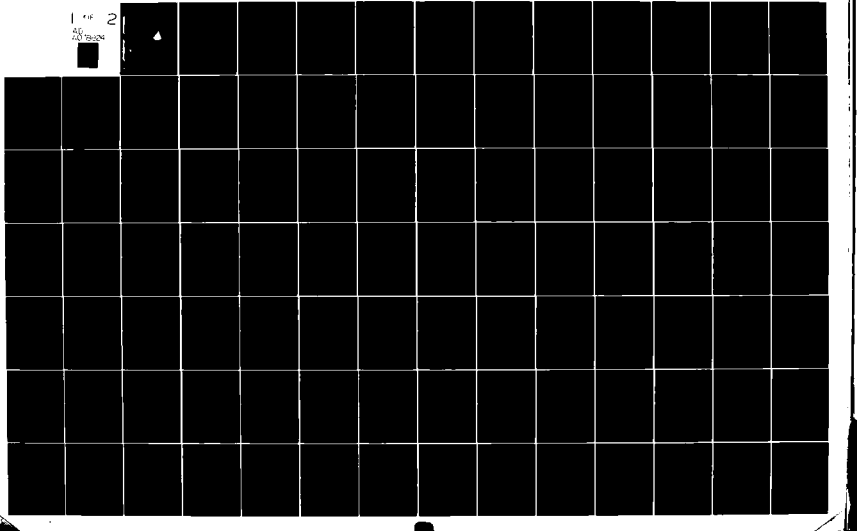
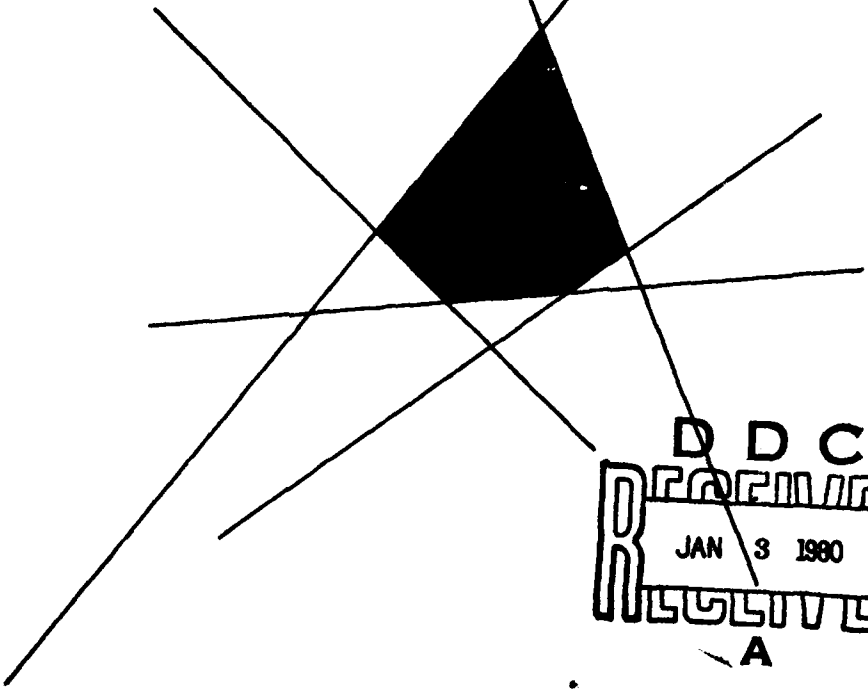
AD-A078 824 CALIFORNIA UNIV BERKELEY OPERATIONS RESEARCH CENTER F/G 12/2
OPTIMAL STATE DETECTION POLICIES FOR COHERENT SYSTEMS.(U)
NOV 77 Y BEN-DOV N00014-75-C-0781
UNCLASSIFIED ORC-77-30 NL



ORC 77-30
NOVEMBER 1977

OPTIMAL STATE DETECTION POLICIES FOR COHERENT SYSTEMS

by
YOSI BEN-DOV



OPERATIONS RESEARCH CENTER

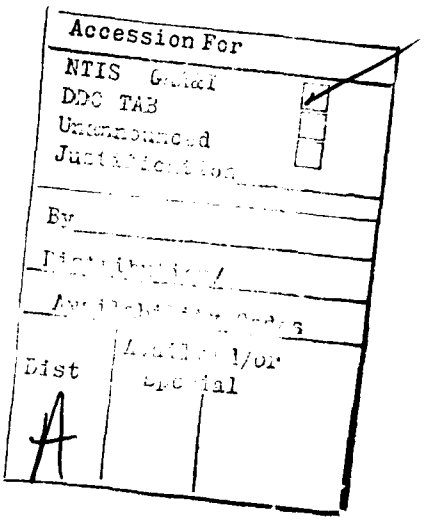
UNIVERSITY OF CALIFORNIA · BERKELEY

OPTIMAL STATE DETECTION POLICIES

FOR COHERENT SYSTEMS

by

Yosi Ben-Dov
Operations Research Center
University of California, Berkeley



NOVEMBER 1977

ORC 77-30

This research has been partially supported by the Air Force Office of Scientific Research (AFSC), USAF, under Grant AFOSR-77-3179 and the Office of Naval Research under Contract N00014-75-C-0781 with the University of California. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: ORC-77-30 / 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): OPTIMAL STATE DETECTION POLICIES FOR COHERENT SYSTEMS.	5. TYPE OF REPORT & PERIOD COVERED: Research Report,
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Yosi Ben-Dov	8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0781 / AFOSR-77-3179
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Operations Research Center, University of California, Berkeley, California 94720	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 2304/A5
11. CONTROLLING OFFICE NAME AND ADDRESS: United States Air Force, Air Force Office of Scientific Research, Bolling AFB, D.C. 20332	12. REPORT DATE: Nov 1977 / 13. NUMBER OF PAGES: 102
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 103	15. SECURITY CLASS. (of this report): Unclassified / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Also supported by the Office of Naval Research under Contract N00014-75-C-0781.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Coherent Systems
Detection Policies
Importance of Components

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(SEE ABSTRACT)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DEDICATION

To my parents who guided me down the long path, and to my wife who gave me the final push.

ACKNOWLEDGEMENT

I would like to express my sincere appreciation to Professor Richard E. Barlow for his advice, supervision and constant inspiration throughout the progress of this thesis. Thanks are due also to Professor Stuart Dreyfus for some helpful comments, and to Professor Austin C. Hoggatt for his critical review of this report. A special vote of thanks to my friends: Alex, Howard, Jan, Jeremy, Ravi, Sancho and Terry who have each added a bit here and there.

Finally, to Gloria Partee for helping me to bring this work into its final form, and to all the other kind people in the Operations Research Center: Fumi, Noreen and Shirley.

ABSTRACT

The problem of minimizing the expected cost of identifying the state of a coherent system (as "functioning" or "failed") is considered. The system is composed of components, and only individual components can be tested. Efficient algorithms are presented for some special cases of coherent systems: series, parallel, parallel-series, series-parallel and k-out-of-n systems. The concept of the Importance of Components is used to develop a branch and bound algorithm which determines the optimal testing procedure for any general coherent system. However this algorithm is not always efficient in solving this problem. Some other closely related problems are discussed, such as how to identify the state of a system which is represented as a fault tree, or how to identify the failed components when the system is known to be failed.

TABLE OF CONTENTS

0. PREFACE

We are interested in the problem of minimizing the expected cost (or time) of testing a coherent system. Our system is composed of n components that either work or fail, and its structure function is monotonically increasing in each argument. By "testing" a system we mean determining its state which can be either "functioning" or "failed." We do the actual testing by checking the components of the system in an optimal sequence. Components can be individually tested and tests give perfect information. Associated with each component is a cost (or time) for testing it $(c_i\ ,\ i = 1,2,\ \ldots,\ n)$ and an a priori probability that it is functioning $(p_i\ ,\ i = 1,2,\ \ldots,\ n)$. (The a priori probability that a component is failed is given by $q_i = 1 - p_i\ ,\ i = 1,2,\ \ldots,\ n$). Components are assumed to function or fail independently of each other. The problem is, therefore, to find the optimal testing policy (which can be represented as a search tree).

One way of dealing with this problem, is by using the techniques of dynamic programming. Let C be the optimal cost of testing a system with n components. $C(i)$, $C(i')$ be the optimal costs of testing a system with $(n - 1)$ components, given that the i^{th} component is working or failed respectively. Then, the dynamic equation for the optimal testing cost is given by:

$$C = \min_i \{c_i + p_i C(i) + (1 - p_i)C(i')\} .$$

It is possible to use this formula recursively, and get the minimum cost of testing any coherent system with n components. The problem is that in order to do so, we have to specify all the different structures of coherent systems for any order. This is an impossible task, since even for small order of n (the number of components in the system) the number of different coherent systems is very large. For example, for $n = 4$ there exist 20 distinct structures of coherent systems, and for $n = 5$ there are 180 structures. For higher orders of n, we do not even know the exact number of different coherent structures, so it is practically impossible to use the dynamic programming formulation to solve this problem.

R. Butterworth [10] solved this problem for the special cases of series and parallel systems, and gave a sufficient condition for the optimality of his procedure for k-out-of-n systems (such a system works if and only if at least k of its n components work). J. Halpern [12] presented an optimal sequential procedure for k-out-of-n systems with equal testing costs for all components, and discussed some other cases [13]. H. Lambert [14], [15] dealt with the problem of minimizing the expected time of testing a failed system. He suggested a criteria for solving the problem for general systems, and showed how to use it specifically for the cases of series and parallel systems. He also conjectured that this criterion gives a good approximation to the optimal procedure for reliable systems.

In the remainder of the preface we give examples to demonstrate possible applications of the different procedures that will be found. These examples will also be used to motivate the solutions, which will be presented in the following chapters of this dissertation.

In Chapter 1 we give an introduction to Reliability Theory, and specifically to the theory of coherent systems. Chapter 2 deals with some special testing procedures that can be applied to special structures of coherent systems. In particular, we present optimal procedures for parallel-series and series-parallel systems, and also for the general case of k-out-of-n systems. A Branch-and-Bound algorithm is presented in Chapter 3, which solves the problem of minimizing the expected cost (time) of testing any general coherent system. The algorithm is based on a paper by Little et al. [18], which gave a branch and bound algorithm for the Travelling Salesman problem. In Chapter 4, other, as yet unsolved, problems are discussed. We consider the problem of minimizing the expected cost of finding the failed components, given that the system has failed, an application to Fault Tree Analysis, and how to get better procedures than the given branch and bound algorithm, for general coherent systems. Further research is needed in those areas, as well as in some other modifications of our problem that we think can be solved in the near future.

Example 1: Communication Network

Consider a hypothetical communication system that may exist between the White House in Washington and the Kremlin in Moscow. There are three ways in which the President of the United States can reach the Chairman of the Communist Party in Russia: either by a direct telephone cable line (line #1), or by a direct telephone satellite line (line #2), or by a telephone cable which goes through the French Capital in Paris (lines #3 and #4). If none of these three ways are available, there exist some non-conventional ways of getting a message to the Russians, which we do not consider. The probability of each of the four communication lines to be in an operating condition when required is given, together with the time it takes to test the condition of each one of the lines as follows:

Line Number:	1	2	3	4
Time for Testing:	30	15	12	10
Probability of Functioning:	.5	.25	.6	.5

What should be the optimal testing procedure, that will minimize the expected time of determining whether the conventional system can transfer a call or not.

Example 2: Setting Admissions Policies

Consider the Graduate Department of Operations Research which needs to set new standards of admitting graduate students. Since the admission committee of the department

decided to raise the admission requirements, it decided to take the following steps in order to reach a decision for students that have already passed the general requirements of the College of Engineering:

Step 1:

Set an interview with three of the professors in the department. Accept any applicant who satisfies the requirements of the 3 professors. Otherwise proceed to Step 2.

Step 2:

Ask for the G.P.A. of the applicant from the science courses during the last two years of his undergraduate studies. Reject any student whose G.P.A. is less than or equal to 3.0. Otherwise, if the G.P.A. is higher than 3.0 proceed to Step 3.

Step 3:

Ask for the G.R.E. scores of the student. Accept any student whose score is higher than or equal to the 80 percentile of the test. Otherwise proceed to Step 4.

Step 4:

Ask for the G.M.A.T. scores of the student. Accept any student whose score is higher than the 90 percentile of the test, and reject all the other students.

The different times that are required to perform each of the above steps are given, and also the probability

of success in any of the tests - as computed from past experience and given as follows:

Test Number:	1	2	3	4
Time of Completion:	30	15	12	10
Probability of Success:	.5	.25	.6	.5

The chairman of the committee would like to know the optimal testing procedure which will minimize the expected time until the committee reaches its decision to accept or reject an applicant.

Example 3:

A power generating system is composed of two identical subsystems which are connected in series. Each of the subsystems has five generators, and the subsystems are considered to be working if at least three out of the five generators work. When the system is failed, the state of any of the subsystems is unknown. Given the different costs of testing each one of the generators and the probability of each one to be functioning as:

Generator Number:	1	2	3	4	5
Cost of Testing:	2	2.5	2	4	3
Probability of Functioning:	.95	.9	.7	.82	.6

What is the optimal testing order which minimizes the expected cost of testing each one of the subsystems?

1. INTRODUCTION TO COHERENT SYSTEM THEORY

1.0 Introduction

In this chapter we present the basic definitions and results of coherent system theory. Most of this work was done in the early 60's by R. E. Barlow, Z. W. Birnbaum, J. D. Esary, F. Proschan and S. C. Saunders at the University of Washington and the Boeing Scientific Research Laboratories [1], [6], [7], [11].

The structural properties of coherent systems are presented, as well as some properties of the reliability function. We emphasis the representation of coherent systems in their parallel-series and series-parallel forms, and the different definitions for the importance of the components in the system.

1.1 The Structure of Coherent Systems

It will be assumed, throughout this work, that any component or any system can only be in one of two possible states: functioning or failed. Given a set of components: $C = \{1,2, \ldots, n\}$ we define a binary indicator variable of component i to be:

$$x_i = \begin{cases} 1 & \text{if component } i \text{ is functioning} \\ 0 & \text{if component } i \text{ is failed;} \end{cases}$$

$\underline{x} = (x_1, \ldots, x_n)$ will be called the "state vector."

1.1.1 Definitions

A *system* is a set of components together with a structure function $\phi(\underline{x})$ (i.e., it can be described as the pair (C,ϕ)), which is determined completely by the states of its components such that:

$$\phi(\underline{x}) = \begin{cases} 1 & \text{if the system is functioning} \\ 0 & \text{if the system is failed.} \end{cases}$$

(C,ϕ) is defined to be a *coherent system* if it satisfies:

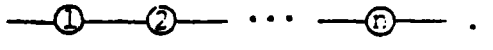
(i) $\underline{x} \leq \underline{y} \Rightarrow \phi(\underline{x}) \leq \phi(\underline{y})$

(ii) $\phi(\underline{0}) = 0 \; ; \; \phi(\underline{1}) = 1$

where: $\underline{0} = (0,0, \ldots, 0)$; $\underline{1} = (1,1, \ldots, 1)$ and $\underline{x} \leq \underline{y}$ when $x_i \leq y_i$ for $i = 1, \ldots, n$. Also: $(1_k,\underline{x}) = (x_1, \ldots, x_{k-1},1_k,x_{k+1}, \ldots, x_n)$.

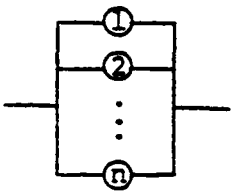
1.1.2 Examples

(i) A *series system*. This system works if and only if all of its components work. It is described by: $\phi(\underline{x}) = \prod_{i=1}^{n} x_i = \min(x_1, \ldots, x_n)$. Its graphical description:



(ii) A *parallel system*. This system fails if and only if all its components fail. This structure is described by: $\phi(\underline{x}) = 1 - \prod_{i=1}^{n} (1 - x_i) = \max(x_1, \ldots, x_n)$.

Its graphical description:



(iii) *k-out-of-n system*. This system works if at least k out of its n components work (and thus fails if at least n-k+1 out of its n components fail). Its structure function:

$$\phi(\underline{x}) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} x_i \geq k \\ 0 & \text{if } \sum_{i=1}^{n} x_i < k \end{cases} .$$

1.1.3 Properties of Coherent Structures

(1.1) (i) $$\phi(\underline{x}) = x_i\phi(1_i,\underline{x}) + (1 - x_i)\phi(0_i,\underline{x})$$

for every $\underline{x}$ $(i = 1,2, \ldots, n)$

i.e., we can express a structure function of order n in terms of a structure function of order $n - 1$.

(1.2) (ii) $$\prod_{i=1}^{n} x_i \leq \phi(\underline{x}) \leq 1 - \prod_{i=1}^{n} (1 - x_i)$$

i.e., the performance of any coherent system is bounded above by the performance of a parallel system with the same components, and bounded below by the performance of a series system with the same components.

(1.3) (iii) a) $$\phi[1 - (1 - \underline{x})(1 - \underline{y})] \geq 1 - [1 - \phi(\underline{x})][1 - \phi(\underline{y})]$$

and:

(1.4) b) $$\phi(\underline{x} \cdot \underline{y}) \leq \phi(\underline{x}) \cdot \phi(\underline{y}) \ .$$

(a) implies that redundancy at the component level is more efficient than redundancy at the system level, a well known principle among engineers.

1.1.4 The Dual Structure

For a given structure function, $\phi(\underline{x})$, we define: $\phi^D(\underline{x}) = 1 - \phi(1 - \underline{x})$, and call it the *dual structure* of ϕ . It is easily shown that: $(\phi^D)^D = \phi$.

Examples:

(i) The dual of a series system is a parallel system.

(ii) The dual of a k-out-of-n system is a (n-k+1)-out-of-n system whose structure function is:

$$\phi(\underline{x}) = \begin{cases} 1 & \sum_{i=1}^{n} x_i \geq n - k + 1 \\ 0 & \sum x_i < n - k + 1 \ . \end{cases}$$

1.1.5 Definitions

A state vector $\underline{x}$ for which $\phi(\underline{x}) = 1$ is called a *path* vector.

A state vector $\underline{x}$ for which $\phi(\underline{x}) = 0$ is called a *cut* vector.

From the first property of coherent systems (the monotonicity property) it follows immediately that:

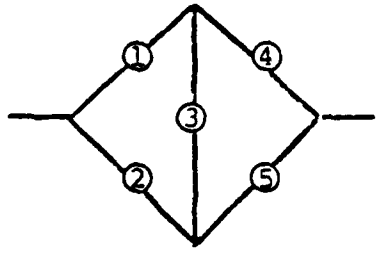
(i) If $\underline{x}$ is a path vector for ϕ and $\underline{y} \geq \underline{x}$ then $\underline{y}$ is a path vector for ϕ . $\underline{x}$ is a *minimal path* vector if $\underline{y} < \underline{x} \Rightarrow \phi(\underline{y}) = 0$.

(ii) If $\underline{x}$ is a cut vector for ϕ and $\underline{y} \leq \underline{x}$ then $\underline{y}$ is a cut vector for ϕ . $\underline{x}$ is a *minimal cut* vector if $\underline{y} > \underline{x} \Rightarrow \phi(\underline{y}) = 1$.

For every path vector and cut vector we define the corresponding path set and cut set as the set of components such that: $C_1(\underline{x}) = \{i \mid x_i = 1\}$ and $C_0(\underline{x}) = \{i \mid x_i = 0\}$ respectively.

Example:

Consider the following "bridge" diagram:



Its min-path sets are $R_1 = \{1,4\}$; $R_2 = \{2,5\}$; $R_3 = \{1,3,5\}$; $R_4 = \{2,3,4\}$. Its min-cut sets are $K_1 = \{1,2\}$; $K_2 = \{4,5\}$; $K_3 = \{1,3,5\}$; $K_4 = \{2,3,4\}$.

1.1.6 Relations Between the Structure and its Dual

(i) If $\underline{x}$ is a path vector for ϕ , then $\underline{1} - \underline{x}$ is a cut vector for ϕ^D , and vice versa.

(ii) The minimal path (cut) sets of the structure ϕ , are the minimal cut (path) sets of its dual ϕ^D .

(iii) If $\phi(\underline{x}) = \phi^D(\underline{x})$ for every $\underline{x}$, then each minimal path set of the structure $\phi(\underline{x})$ is also a minimal cut set of $\phi(\underline{x})$. Those systems are called *self-dual* systems.

1.1.7 Representation of Coherent Systems

A coherent system works if and only if all the components in at least one minimal path set work. A coherent system fails if and only if all the components in at least one minimal cut set fail. Using these facts we can write the structure function of any coherent system as a parallel arrangement of its minimal paths or as a series arrangement of its minimal cuts.

Specifically, let: $\mathbb{R} = \{R_1, \ldots, R_r\}$ be the set of all minimal path sets of the system, together with their series structures: $\rho_1, \ldots, \rho_r$ such that: $\rho_j(\underline{x}) = \prod_{i \in R_j} x_i$.

Let: $\mathbb{K} = \{K_1, \ldots, K_k\}$ be the minimal cut sets of the system with their parallel structures: $\psi_1, \ldots, \psi_k$ such that: $\psi_j = 1 - \prod_{i \in K_j} (1 - x_i)$. Then: $\phi(\underline{x}) = 1 - \prod_{j=1}^{r} [1 - \rho_j(\underline{x})]$ where: $\rho_j(\underline{x}) = \prod_{i \in R_j} x_i$, $j = 1,2, \ldots, r$ and $\phi(\underline{x}) = \prod_{j=1}^{k} \psi_j(\underline{x})$ where: $\psi_j(\underline{x}) = 1 - \prod_{i \in K_j} (1 - x_i)$, $j = 1,2, \ldots, k$. These equations are called the parallel-series and the series-parallel representation, respectively.

Definition:

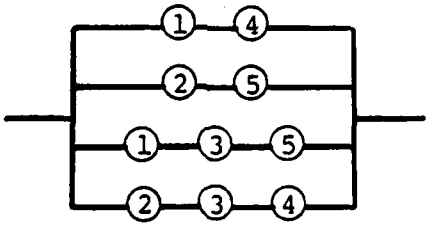
A *parallel-series* system is a coherent system which can be described as a parallel arrangement of *disjoint* series subsystems, i.e.: $\phi(\underline{x}) = 1 - \prod_{j=1}^{r} (1 - \rho_j(\underline{x}))$ where: $\rho_j(\underline{x}) = \prod_{i \in R_j} x_i$ and: $R_j \cap R_i = \phi$ for every pair: $i \neq j$, $1 \leq i, j \leq r$.

Definition:

A *series-parallel* system is a coherent system which can be described as a series sequence of *disjoint* parallel subsystems, i.e.: $\phi(\underline{x}) = \prod_{j=1}^{k} \psi_j(\underline{x})$ where: $\psi_j(\underline{x}) = 1 - \prod_{i \in K_j} (1 - x_i)$ and: $K_j \cap K_i = \phi$ for every pair $1 \leq i, j \leq k, j \neq i$.

Examples:

(i) The "bridge" structure, from Section 1.1.5. We have already found the minimal path and the minimal cut sets of this structure. The parallel-series representation will be:



where:

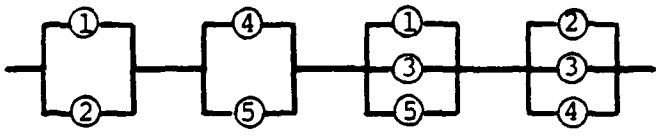
$$\rho_1(\underline{x}) = x_1 x_4$$

$$\rho_2(\underline{x}) = x_2 x_5$$

$$\rho_3(\underline{x}) = x_1 x_3 x_5$$

$$\rho_4(\underline{x}) = x_2 x_3 x_4$$

The series-parallel representation will be:



where:

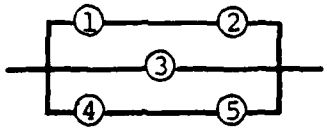
$$\psi_1(\underline{x}) = 1 - (1 - x_1)(1 - x_2)$$

$$\psi_2(\underline{x}) = 1 - (1 - x_4)(1 - x_5)$$

$$\psi_3(\underline{x}) = 1 - (1 - x_1)(1 - x_3)(1 - x_5)$$

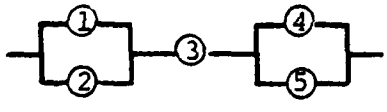
$$\psi_4(\underline{x}) = 1 - (1 - x_2)(1 - x_3)(1 - x_4)$$

(ii) The following system:



$$\phi(\underline{x}) = 1 - (1 - x_1x_2)(1 - x_3)(1 - x_4x_5)$$

has 3 minimal path sets: $R_1 = \{1,2\}$; $R_2 = \{3\}$; $R_3 = \{4,5\}$. Since $R_1 \cap R_2 = R_1 \cap R_3 = R_2 \cap R_3 = \phi$ this is a parallel-series system. Its dual system:



$$\phi^D(\underline{x}) = [1 - (1 - x_1)(1 - x_2)]x_3[1 - (1 - x_4)(1 - x_5)]$$

has 3 minimal cut sets: $K_1 = \{1,2\}$; $K_2 = \{3\}$; $K_3 = \{4,5\}$, and since: $K_1 \cap K_2 = K_1 \cap K_3 = K_2 \cap K_3 = \phi$ this is a series-parallel system. Note that the minimal path sets of the dual system: $R_1 = \{1,3,4\}$; $R_2 = \{1,3,5\}$; $R_3 = \{2,3,4\}$; $R_4 = \{2,3,5\}$ which are equal to the minimal cut sets of the original system, do not satisfy the requirement that the intersection between any pair is empty. (e.g., $R_1 \cap R_2 = \{1,3\}$ etc.)

1.1.8 Structural Importance of Components

Definition:

A path vector $(1_i, \underline{x})$ is said to be *critical* for component i if it satisfies:

$$\phi(1_i, \underline{x}) - \phi(0_i, \underline{x}) = 1 \ .$$

Let: $n_\phi(i) = \sum_{\{x | x_i = 1\}} [\phi(1_i, \underline{x}) - \phi(0_i, \underline{x})]$ be the total number of critical path vectors for component i .

Definition:

A measure of the structural importance of component i is given by:

$$I_\phi(i) = \frac{1}{2^{n-1}} n_\phi(i) = \frac{1}{2^{n-1}} \sum_{\{x | x_i = 1\}} [\phi(1_i, \underline{x}) - \phi(0_i, \underline{x})] \ .$$

$I_\phi(i)$ is the proportion of the 2^{n-1} vectors of the form $(1_i, \underline{x})$ which are critical for component i.

Note:

(i) $I_\phi(j) = 0 \Rightarrow$ component j is irrelevant to the system.

(ii) $I_\phi(k) = 1 \Rightarrow$ component k is the only relevant component in the system, i.e., the system works if and only if component k works.

In general, $0 \leq I_\phi(i) \leq 1$.

Examples:

(i) *Series system* (with n components). Since $\phi(\underline{x}) = \prod_{i=1}^{n} x_i = 1$ only when: $\underline{x} = (1,1, \ldots, 1)$,
$n_\phi(i) = 1$ for every component, i.e.;
$I_\phi(i) = \frac{1}{2^{n-1}}$ $i = 1,2, \ldots, n$.

(ii) *Parallel system* (with n components). Since $\phi(\underline{x}) = 1 - \prod_{i=1}^{n} (1 - x_i)$ then only the vector $(0, \ldots, 0, 1_i, 0, \ldots, 0)$ is critical for the i-th component, i.e.; $I_\phi(i) = \frac{1}{2^{n-1}}$
$i = 1,2, \ldots, n$.

(iii) *k-out-of-n system*. In this example, $n_\phi(i) = \binom{n-1}{k-1}$ for $i = 1,2, \ldots, n$ since given that $x_i = 1$, we have $\binom{n-1}{k-1}$ possibilities for

exactly $(k - 1)$ additional components to be equal 1, as needed for any critical path vector for a k-out-of-n system. Therefore:

$$I_{\phi}(i) = \frac{\binom{n-1}{k-1}}{2^{n-1}} = \frac{(n-1)!}{(k-1)!(n-k)!2^{n-1}}$$

$$i = 1,2, \ldots, n .$$

Conclusion:

For symmetric systems like the series, parallel and k-out-of-n systems, all the components are equally important as far as the structural importance is concerned.

1.1.9 Modules of Coherent Systems

Definition:

The coherent system (A,χ) is a *module* of the coherent system (C,ϕ) if: $\phi(\underline{X}) = \psi\left[\chi(\underline{X}^A), \underline{X}^{A^C}\right]$ where ψ is a coherent structure, $A \subseteq C$ and $A^C = C - A$. $A \subseteq C$ is a modular set of C. If $A \subset C$ it is called a proper modular, and (A,χ) is a proper module of (C,ϕ).

Definition:

A *modular decomposition* of (C,ϕ) is a set of disjoint modules: $\{(A,\chi_1),(A_2,\chi_2), \ldots, (A_r,\chi_r)\}$ together with an organizing structure function ψ such that:

i) $C = \bigcup_{i=1}^{r} A_i$ where: $A_i \cap A_j = \phi$ for $i \neq j$.

ii) $\phi(\underline{x}) = \psi\left[\chi_1\left(\underline{x}^{A_1}\right), \chi_2\left(\underline{x}^{A_2}\right), \ldots, \chi_r\left(\underline{x}^{A_r}\right)\right]$.

Example:

Consider the system in 1.1.7, Example (ii). Define the modular sets: $A_1 = \{1,2,3\}$; $A_2 = \{4,5\}$. Define the structure functions:

$$\chi_1 = 1 - [(1 - x_1x_2)(1 - x_3)] \; ; \; \chi_2 = x_4x_5 \; .$$

Then $\{(A,\chi_1);(A_2,\chi_2)\}$ is a modular decomposition of the given system because:

i) $C = A_1 \cup A_2 = \{1,2,3,4,5\}$

ii) $\phi(\underline{x}) = 1 - \left[1 - \chi_1\left(\underline{x}^{A_1}\right)\right]\left[1 - \chi_2\left(\underline{x}^{A_2}\right)\right]$

$= 1 - [(1 - x_1x_2)(1 - x_3)(1 - x_4x_5)]$.

1.2 The Reliability of Coherent Systems

1.2.1 Definition of Reliability Function

Let ϕ be a structure of order n . Assume that the indicator variables of its n components, are independent random variables, X_i such that: $\Pr\{X_i = 1\} = p_i$; $\Pr\{X_i = 0\} = 1 - p_i = q_i$, $i = 1,2, \ldots, n$.

Definition:

The *reliability of a component* is equal to its probability of working, p_i .

Since $X_1, X_2, \ldots, X_n$ are random variables, $\phi(\underline{X}) = \phi(X_1, X_2, \ldots, X_n)$ is also a random variable, where $\underline{X} = (X_1, \ldots, X_n)$.

Definition:

The *reliability of a system* is the probability that the system works, which is given by

$$\Pr\{\phi(\underline{X}) = 1\} = h = E[\phi(\underline{X})] .$$

Note: Since the indicators are independent random variables, we can write the reliability function, h , as: $h = h(\underline{p})$, where $\underline{p} = (p_1, p_2, \ldots, p_n)$.

1.2.2 Examples

We compute the reliability functions for the systems given in 1.1.2:

(i) A *series system*. $\phi(\underline{x}) = \prod_{i=1}^{n} x_i$ and:

$$h(\underline{p}) = \prod_{i=1}^{n} p_i .$$

(ii) A *parallel system*. $\phi(\underline{x}) = 1 - \prod_{i=1}^{n} (1 - x_i)$.

The reliability function is given by:

$$h(\underline{p}) = 1 - \prod_{i=1}^{n} (1 - p_i) .$$

(iii) A *k-out-of-n* system. Recall that:

$$\phi(\underline{x}) = \begin{cases} 1 & \sum_{i=1}^{n} x_i \geq k \\ 0 & \sum_{i=1}^{n} x_i < k \end{cases} .$$ The reliability

function of a k-out-of-n system, with n identical components, is: $h(\underline{p}) = \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i}$.

1.2.3 General Properties of Reliability Functions

(1.5) (i) $h(\underline{p}) = p_i h(1_i, \underline{p}) + (1 - p_i) h(0_i, \underline{p})$ for $i = 1,2, \ldots, n$.

Proof:

From (1.1) recall that:

$$\phi(\underline{x}) = x_i \phi(1_i, \underline{x}) + (1 - x_i)\phi(0_i, \underline{x}) \quad \text{for every } \underline{x}$$
$$i = 1,2, \ldots, n .$$

Also:

$$h(\underline{p}) = E[\phi(\underline{X})] = E(X_i) \cdot E[\phi(1_i, \underline{X})] + [1 - E(X_i)]E[\phi(0_i, \underline{X})]$$

and from the independence of the components the result follows immediately. ■

(ii) $h(\underline{0}) = 0$ and $h(\underline{1}) = 1$, which is a direct result from the definition of coherent systems.

(iii) $h'(\underline{p}) > 0$ for $0 < \underline{p} < 1$ i.e.; the reliability of the structure strictly increases as the components' reliabilities increase.

Proof:

$$\frac{\partial h}{\partial p_i} = h(1_i, \underline{p}) - h(0_i, \underline{p}) = E[\phi(1_i, \underline{X}) - \phi(0_i, \underline{X})]$$

and the result follows from the properties of the structure function.■

1.2.4 Reliability Importance of Components

The measure of Reliability Importance takes into account the structural importance of the components as well as their failure probabilities. This measure tells us how much we can improve the system reliability by improving the reliability of any single component. Specifically, we shall use Birnbaum's definition of importance:

Definition:

The Reliability Importance I_j of component j is given by:

$$I_j = \frac{\partial h(\underline{p})}{\partial p_j} .$$

Using the identity (1.5):

$$h(\underline{p}) = p_i h(1_i,\underline{p}) + (1 - p_i)h(0_i,\underline{p}) \quad \text{for} \quad i = 1,2, \ldots, n$$

we can get an equivalent definition for I_j to be:

$$(1.6) \qquad I_j = \frac{\partial h(\underline{p})}{\partial p_j} = h(1_j,\underline{p}) - h(0_j,\underline{p})$$

or more explicitly:

$$(1.7) \quad I_j = E[\phi(1_j,\underline{X}) - \phi(0_j,\underline{X})] = P\{[\phi(1_j,\underline{X}) - \phi(0_j,\underline{X})] = 1\} .$$

Remarks:

(i) If the components satisfy the conditions: $0 < p_i < 1$ for every i, then: $0 < I_j < 1$ for every i.

Proof:

Since the structure function ϕ is increasing, $\phi(1_i,\underline{X}) - \phi(0_i,\underline{X}) \geq 0$. Also, the definition of coherent system requires that each component is relevant, i.e.; $\phi(1_i,\underline{X}^o) - \phi(0_i,\underline{X}^o) = 1$ for some $\underline{X}^o$. Finally, since $0 < p_i < 1$ for every i, $\underline{X}^o$ has a positive probability of occurrence and hence: $E[\phi(1_i,\underline{X}) - \phi(0_i,\underline{X})] > 0$. Since $E[\phi(1_i,\underline{X}) - \phi(0_i,\underline{X})] < 1$ it follows that $0 < I_j < 1$. ■

(ii) If $p_i = \frac{1}{2}$ for every i, then: $I_j = \left.\frac{\partial h(\underline{p})}{\partial p_i}\right|_{p_1=p_2=\cdots=p_n=\frac{1}{2}}$ which gives us Birnbaum's definition of Structural Importance of the components, i.e.;

$$I_\phi(j) = \frac{1}{2^{n-1}} \sum_{\{x|x_j=1\}} [\phi(1_j,\underline{x}) - \phi(0_j,\underline{x})] .$$

Note that $\sum_{\{x|x_j=1\}} [\phi(1_j,\underline{X}) - \phi(0_j,\underline{X})]$ is the total number of state vectors in which component j is critical to the system, in the sense that for those vectors the system will be functioning if and only if component j will be functioning.

Examples:

For the following examples assume that the components of the system have been labeled such that their probabilities are ordered as:

$$p_1 \leq p_2 \leq \cdots \leq p_n .$$

(i) *Series System.* The reliability function of a series system is given by: $h(\underline{p}) = \prod_{i=1}^{n} p_i$ and thus: $I_j = \prod_{i\neq j} p_i$. Therefore: $I_1 \geq I_2 \geq \cdots \geq I_n$ i.e., the least reliable component is the most important in the system. A series system is functioning if and only if all of its components are functioning, so that the failure of any component causes a system failure, and the system is no better than its weakest component. Thus, the weakest component in the system is the most important component.

(ii) *Parallel System.*

$$h(\underline{p}) = 1 - \prod_{i=1}^{n} (1 - p_i)$$

thus: $I_j = \prod_{i \neq j} (1 - p_i)$. Therefore:

$I_1 \leq I_2 \leq \cdots \leq I_n$ i.e., the most reliable component is the most important in the system. This, again, is very intuitive, since a parallel system functions if at least one of its components functions - thus, the strongest component in the system is the most important.

(iii) *2-out-of-3 System.* The reliability function of a 2-out-of-3 system is given by:

$h(\underline{p}) = p_1p_2 + p_1p_3 + p_2p_3 - 2p_1p_2p_3$, and thus:

$$I_1 = p_2 + p_3 - 2p_2p_3$$

$$I_2 = p_1 + p_3 - 2p_1p_3$$

$$I_3 = p_1 + p_2 - 2p_1p_2 \cdot$$

If $p_i = \frac{1}{2}$ for $i = 1,2,3$, then: $I_1 = I_2 = I_3$, since all the components are equally important as far as the structure of the system is concerned.

If $p_i \leq \frac{1}{2}$ for $i = 1,2,3$, then: $I_1 \geq I_2 \geq I_3$, i.e., the weakest component is the most important for the system.

If $p_i \geq \frac{1}{2}$ for $i = 1,2,3$, then $I_3 \geq I_2 \geq I_1$, i.e., the most important component is the one with the highest reliability.

Otherwise, $I_2 \geq \max \{I_1, I_3\}$, i.e., the most important component is the one whose probability of working is neither the highest nor the smallest among the three components.

2. OPTIMAL TESTING PROCEDURES FOR SPECIAL STRUCTURES OF COHERENT SYSTEMS

2.0 Introduction

In this chapter we consider the problem of determining the order for testing components which will determine the state of the system at minimum expected cost (or minimum expected time). For series and parallel systems, Butterworth [10] gave the optimal procedures which we present in Sections 1 and 2. We generalize his results for parallel-series and series-parallel systems in Sections 3 and 4. Halpern [13] discussed this problem when the costs of testing are equal for all the components. Finally, in Section 5, we give an optimal procedure for k-out-of-n systems. Halpern [12] dealt with a special case of these systems.

2.1 Series Systems

Testing a series system is a sequential process, and the need to make the i^{th} test in the system, implies that the first $i - 1$ tests found all the components to be working. The expected cost of testing a series system is therefore given by:

$$C(\pi) = c_{\pi_1} + \sum_{i=2}^{n} \left[\prod_{j=1}^{i-1} p_{\pi_j} \right] c_{\pi_i} , \tag{2.1}$$

where $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ is a permutation of the components which corresponds to the testing order.

2.1.1 Lemma

The procedure $\pi = (1,2, \ldots, n)$ is optimal for any series system, if and only if: $\frac{c_1}{q_1} \leq \frac{c_2}{q_2} \leq \cdots \leq \frac{c_n}{q_n}$.

Proof:

Order components so that $\frac{c_1}{q_1} \leq \frac{c_2}{q_2} \leq \cdots \leq \frac{c_n}{q_n}$. We wish to show that $\pi = (1,2, \ldots, n)$ is the optimal procedure.

Suppose there exists another procedure, π', which is better, i.e., $C(\pi') < C(\pi)$. Assume also that π' differs from π only by a single interchange of k and $k+1$. By expanding the expressions for $C(\pi)$ and $C(\pi')$ we will get:

$$C(\pi) = c_1 + \sum_{i=2}^{k-1} \left[\prod_{j=1}^{i-1} p_j\right] c_i + \prod_{j=1}^{k-1} p_j c_k + \prod_{j=1}^{k-1} p_j p_k c_{k+1} + \sum_{i=k+2}^{n} \left[\prod_{j=1}^{i-1} p_j\right] c_i$$

$$C(\pi') = c_1 + \sum_{i=2}^{k-1} \left[\prod_{j=1}^{i-1} p_j\right] c_i + \prod_{j=1}^{k-1} p_j c_{k+1} + \prod_{j=1}^{k-1} p_j p_{k+1} c_k + \sum_{i=k+2}^{n} \left[\prod_{j=1}^{i-1} p_j\right] c_i$$

but:

$$C(\pi) > C(\pi') \iff \prod_{j=1}^{k-1} p_j c_k + \prod_{j=1}^{k-1} p_j p_k c_{k+1} > \prod_{j=1}^{k-1} p_j c_{k+1} + \prod_{j=1}^{k-1} p_j p_{k+1} c_k$$

$$\iff c_k + p_k c_{k+1} > c_{k+1} + p_{k+1} c_k$$

$$\iff c_k(1 - p_{k+1}) > c_{k+1}(1 - p_k)$$

$$\iff \frac{c_k}{q_k} > \frac{c_{k+1}}{q_{k+1}}$$

which contradicts the assumed ordering.

Since any procedure can be described as a series of single interchanges like the one above, we have proven, in general, that the procedure π is optimal for any series system.■

2.1.2 Remarks

First, we want to note that this result is very intuitive. The component which is being tested first is the one which has the greatest probability for the completion of the test, per unit of testing cost.

Second, it should be noted that this problem was solved in other contexts (Knuth [15]), and it has many applications in a large variety of areas.

2.2 Parallel Systems

For parallel systems we can also find an optimal sequential procedure, which will assure us that we will minimize the expected cost of testing any parallel system. For those systems, checking the i^{th} component implies that the first $i - 1$ components were found to be failed,

otherwise we would have stopped our testing procedure when a working component was found. The expected cost of testing a parallel system is therefore given by:

$$(2.2) \qquad C(\pi) = c_{\pi_1} + \sum_{i=2}^{n} \left[\prod_{j=1}^{i-1} q_{\pi_j} \right] c_{\pi_i} \ .$$

2.2.1 Lemma

The procedure $\pi = (1,2, \ldots, n)$ is optimal for any parallel system, if and only if $\frac{c_1}{p_1} \leq \frac{c_2}{p_2} \leq \cdots \leq \frac{c_n}{p_n}$.

Proof:

Note the duality between the parallel and the series systems, which appears in the corresponding formulas for $C(\pi)$. Again it is easy to see that $\pi = (1, \ldots, n)$ minimizes cost against any other permutation, when $\frac{c_1}{p_1} \leq \frac{c_2}{p_2} \leq \cdots \leq \frac{c_n}{p_n}$.■

2.3 Parallel-Series Systems

Recall from 1.1.7 that parallel-series systems are coherent systems which can be described as a parallel arrangement of disjoint series subsystems. Hence:

$$\phi(\underline{x}) = 1 - \prod_{i=1}^{r} [1 - \rho_i(\underline{x})] = 1 - \prod_{i=1}^{r} \left[1 - \left(\prod_{j \in R_i} x_j \right) \right]$$

where $R_1, \ldots, R_r$ are the minimal path-sets of the system.

In order to define the optimal policy for the parallel-series systems, we first have to order the components in each one of the disjoint min path-sets according to the optimal rule for testing series systems. Let $S_{j1}, S_{j2}, \ldots, S_{jn(j)}$ be the ordered set of components in the j^{th} min path-set.

The expected cost of testing this min path-set is

$$E(R_j) = c_{j,1} + \sum_{i=2}^{n(j)} \left[\prod_{k=1}^{i-1} p_{j,k}\right] c_{j,i} , \tag{2.3}$$

where $n(j)$ are the number of components in R_j. The probability that the j^{th} min path-set is working is given by

$$P(R_j) = \prod_{i=1}^{n(j)} p_{ji} . \tag{2.4}$$

Define $E(R_1), \ldots, E(R_r)$ and $P(R_1), \ldots, P(R_r)$ as above for all the minimal path-sets in the system. By the procedure $\pi = (R_1, R_2, \ldots, R_r)$ for a parallel-series system, we mean test each min path in turn beginning with R_1 and determine its state according to the rule for testing series systems given by Lemma 2.1.1. Testing stops as soon as a min path is found all of whose components are working or after all min paths are found failed.

2.3.1 Theorem

The optimal testing procedure for a parallel-series system always finishes determining the state of a minimal path set before it moves on to test another minimal path set.

The procedure $\pi = (R_1, \ldots, R_r)$ is optimal for any parallel-series system, if and only if

$$\frac{E(R_1)}{P(R_1)} \leq \frac{E(R_2)}{P(R_2)} \leq \cdots \leq \frac{E(R_r)}{P(R_r)} .$$

Proof:

The proof is by induction on the number of components in the system. For $n = 2$, we either have a series system or a parallel system, and the proof is obvious. Now, assume that our theorem is true for a parallel-series system with $n - 1$ components. We will show that it must be true for a parallel-series system with n components.

Suppose the optimal testing procedure, π, for testing an n component parallel-series system first examines a component, S_1, in a min path set labelled R_1. If S_1 is failed then the state of min path set R_1 is known and obviously the optimal procedure moves to another min path set. Since the union of remaining min path sets have less than n components, the state of each of the remaining min paths is always determined before the procedure moves on, by the induction hypothesis.

Now, suppose we first examine S_1 and we find it working. We then have a parallel-series system with $n - 1$ components. By the induction hypothesis, the optimal procedure for this system always determines the state of a min path set before moving on. Hence min path sets are like super components in a parallel system. Label the remaining min path sets $R_2, R_3, \ldots, R_r$ so that

$$\frac{ER_2}{P(R_2)} \leq \frac{ER_3}{P(R_3)} \leq \cdots \leq \frac{ER_r}{P(R_r)}$$

where ER_j is the expected cost of determining the state of R_j (a series system) and $P(R_j)$ is the probability that all components in R_j are working. Let R_1' be the remaining components in R_1 after we have examined S_1 and found it working.

Case 1:

$\frac{ER_1'}{P(R_1')} \leq \frac{ER_2}{P(R_2)}$. By Lemma 2.2.1 and the induction hypothesis, the optimal policy continues to examine components in R_1 until the state of R_1 is determined.

Case 2:

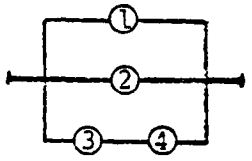
$\frac{ER_1'}{P(R_1')} > \frac{ER_2}{P(R_2)}$. (We show that in this case π is not optimal.) Given S_1 is working, the optimal policy by Lemma 2.2.1 and the induction hypothesis then examines components in R_2. But we can show that π cannot possibly be optimal in this case. Consider the policy π' which first determines the state of R_2 and then examines S_1. Policy π' is equivalent to π since π, in this case, always moves immediately to R_2 regardless of the state of S_1. But π' is obviously not optimal since even if R_2 is working, it requires an additional check of S_1. Hence π is not optimal and we have a contradiction. It follows that

$$\frac{ER_1'}{P(R_1')} > \frac{ER_2}{P(R_2)}$$

cannot hold for the optimal policy, and the optimal policy always determines the state of a min path set before moving on.■

2.3.2 Example

Consider Example 1 in the preface, which can be represented as the following coherent system:



$$\phi(\underline{x}) = 1 - [(1 - x_1)(1 - x_2)(1 - x_3x_4)]$$

i	c_i	p_i
1	30	.5
2	15	.25
3	12	.6
4	10	.5

$R_1 = \{1\}$

$R_2 = \{2\}$

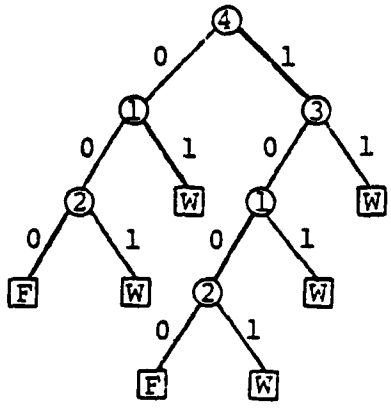
$R_3 = \{4,3\}$

$E(R_1) = 30$ $\quad P(R_1) = .5$ $\quad \frac{E(R_1)}{P(R_1)} = 60$

$E(R_2) = 15$ $\quad P(R_2) = .25$ $\quad \frac{E(R_2)}{P(R_2)} = 60$

$E(R_3) = 10 + .5 \cdot 12 = 16$ $\quad P(R_3) = .3$ $\quad \frac{E(R_3)}{P(R_3)} = 53.33$.

The optimal procedure can be described using the following search tree (the number on the branch determines the state of the tested component to be "functioning" (= 1) or "failed" (= 0), "W" denotes a working system, and "F" stands for a failed system):



The expected cost of testing the system using the optimal procedure is given by:

$$C(\pi) = c_4 + p_4c_3 + (q_4 + p_4q_3)c_1 + (q_4q_1 + p_4q_3q_1)c_2$$

$$= 30 + 6 + 21 + 1.5 = 58.5 \ .$$

2.4 Series-Parallel Systems

Recall from 1.1.7 that series-parallel systems are coherent systems which can be described as a series sequence of disjoint parallel subsystems. Hence:

$$\phi(\underline{x}) = \prod_{i=1}^{k} [\psi_i(\underline{x})] = \prod_{i=1}^{k} \left[1 - \prod_{j \in K_i} (1 - x_j)\right]$$

where: $K_1, K_2, \ldots, K_k$ are the minimal cut-sets of the system.

The optimal search procedure for the series-parallel systems can be found in a dual fashion to the optimal procedure for the parallel-series system. We will omit the proof for the following theorem, since it can be developed exactly like the proof in the previous section.

2.4.1 Theorem

The optimal testing procedure for a series parallel system always finishes determining the state of a minimal cut set, before it moves on to test another minimal cut set.

The actual testing procedure requires first a reordering of the components in each minimal cut set. In this case the ordering will follow the optimal rule for testing parallel systems. Let: $T_{j,1}, T_{j,2}, \ldots, T_{j,m(j)}$ be the ordered set of components in the j^{th} minimal cut set

i.e., $\frac{c_{T_{j,1}}}{p_{T_{j,1}}} \leq \frac{c_{T_{j,2}}}{p_{T_{j,2}}} \leq \cdots \leq \frac{c_{T_{j,m(j)}}}{p_{T_{j,m(j)}}}$. The expected cost of testing this cut set is given by

$$F(K_j) = c_{j,1} + \sum_{i=2}^{m(j)} \left[\prod_{k=1}^{i-1} q_{jk} \right] c_{ji} \cdot \tag{2.5}$$

The probability that this cut set will fail is given by

$$Q(K_j) = \prod_{i=1}^{m(j)} q_{j,i} \quad . \tag{2.6}$$

Let $P(K_j) = 1 - Q(K_j)$, $j = 1,2, \ldots, k$. By the procedure $\pi = (K_1, K_2, \ldots, K_k)$ we mean test min cut sets $K_1, K_2, \ldots, K_k$ in order, determining the state of each min cut set before moving on. Components in min cut set K_j are tested according to the rule for parallel systems given by Lemma 2.2.1. Testing stops as soon as a min cut set is found all of whose components have failed, or after all min cuts are found working.

2.4.2 Theorem

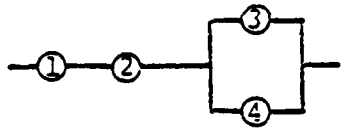
The procedure $\pi = (K_1, \ldots, K_k)$ is optimal for any series-parallel system, if and only if:

$$\frac{F(K_1)}{Q(K_1)} \leq \frac{F(K_2)}{Q(K_2)} \leq \cdots \leq \frac{F(K_k)}{Q(K_k)} \quad .$$

The proof is similar to that of Theorem 2.3.1.

2.4.3 Example

Consider the dual structure for the system that was introduced in the previous section:



where

$$\phi(\underline{x}) = x_1 x_2 [1 - (1 - x_3)(1 - x_4)]$$

i	c_i	p_i
1	30	.5
2	15	.25
3	12	.6
4	10	.5

$K_1 = \{1\}$

$K_2 = \{2\}$

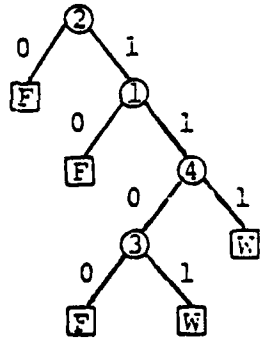
$K_3 = \{3,4\}$

$F(K_1) = 30$ $\quad Q(K_1) = .5$ $\quad \frac{F(K_1)}{Q(K_1)} = 60$

$F(K_2) = 15$ $\quad Q(K_2) = .75$ $\quad \frac{F(K_2)}{Q(K_2)} = 20$

$F(K_3) = 10 + .5 \cdot 12 = 16$ $\quad Q(K_3) = .2$ $\quad \frac{F(K_3)}{Q(K_3)} = 80$.

The "search tree" description of the optimal testing procedure is



The expected cost of testing the system using the optimal procedure, π , is

$$C(\pi) = c_2 + p_2c_1 + p_2p_1c_4 + p_2p_1q_4c_3$$

$$= 15 + 7.5 + 1.25 + .75 = 24.5 .$$

2.5 k-out-of-n Systems

Halpern [12], solved the problem of finding the optimal procedure, when the costs of testing all the components are equal. We present here a generalization of his procedure which can be applied to any k-out-of-n system, with different probabilities and different costs. Since our procedure is adaptive sequential (like the procedure in [12] and unlike the result in [10]), we only have to specify the rule for finding the first component to be tested under the optimal policy. The decision about the next component to be tested is made only after the previous result is already known, at which time we either have a $(k - 1)$-out-of-$(n - 1)$ or a k-out-of-$(n - 1)$ system, depending on whether the previous component was found to be working or failed respectively. This process is continued sequentially until we either get a 1-out-of-ℓ (for some $\ell < n$) system or a ℓ-out-of-ℓ system. These systems are parallel and series respectively, and the optimal testing procedure for these systems is given in Sections 1 and 2.

We follow Halpern's notation and the framework of his arguments. After relabeling the components of the system so that: $\frac{c_1}{p_1} \leq \frac{c_2}{p_2} \leq \cdots \leq \frac{c_n}{p_n}$ we prove

2.5.1 Theorem

For k-out-of-n systems, testing first the component which satisfies

$$\frac{c_i}{q_i} = \min_{1 \leq j \leq k} \left\{ \frac{c_j}{q_j} \right\},$$

results in an optimal testing procedure which minimizes the expected cost of testing over all possible procedures. Testing stops as soon as either k components have been found working or $n - k + 1$ components have been found failed.

Before proving the optimality of this procedure, we give an example.

2.5.2 Example

Consider the 3-out-of-5 system of Example 3 in the preface with the following information about its components:

i	c_i	p_i	c_i/p_i	c_i/q_i
1	2	.95	2.11	40
2	2.5	.9	2.78	25
3	2	.7	2.86	6.67
4	4	.82	4.88	22.22
5	3	.6	5	7.5

Note that we have labeled the components so that $\frac{c_1}{p_1} < \frac{c_2}{p_2} < \cdots < \frac{c_5}{p_5}$.

According to Theorem 2.5.1 the first component to be tested is component 3, since: $\frac{c_3}{q_3} = \min_{i=1,2,3} \left\{\frac{c_i}{q_i}\right\}$. If this component is found to be working, then we have a 2-out-of-4 system, which is composed of components 1,2,4,5. This is true since for a 3-out-of-5 system we need that at least 3 out of the 5 components will be working, for the system to be working. Once we find that we have a working component in our first test (component number 3), we need to find 2 more components (out of the remaining 4) to be working, in order for the system to be working. If, on the other hand, component 3 is found to be failed, we still need 3 components out of the remaining 4 to be found working, in order that the system will be found working. In the first case, we will test component 2 next since $\frac{c_2}{q_2} = \min_{i=1,2} \left\{\frac{c_i}{q_i}\right\}$. In the later case, we will test next component 4 since $\frac{c_4}{q_4} = \min_{i=1,2,4} \left\{\frac{c_i}{q_i}\right\}$. We can demonstrate the final testing procedure by the following search tree (Figure 2.1).

The expected cost of testing the given system using the candidate algorithm is therefore:

$$C = c_3 + [p_3 + q_3(p_4 + q_4p_5)]c_2 + [p_3(q_2 + p_2q_1) + q_3]c_4$$
$$+ [p_3(p_2q_1q_4 + q_2q_4 + q_2p_4q_1) + q_3(q_4 + p_4q_2 + p_4p_2q_1)]c_5$$
$$+ [p_3(p_2 + q_2p_4 + q_2q_4p_5) + q_3(p_4p_2 + p_4q_2p_5 + q_4p_5p_2)]c_1$$
$$= 2 + 2.446 + 1.606 + .33243 + 1.92056 = 8.30499 .$$

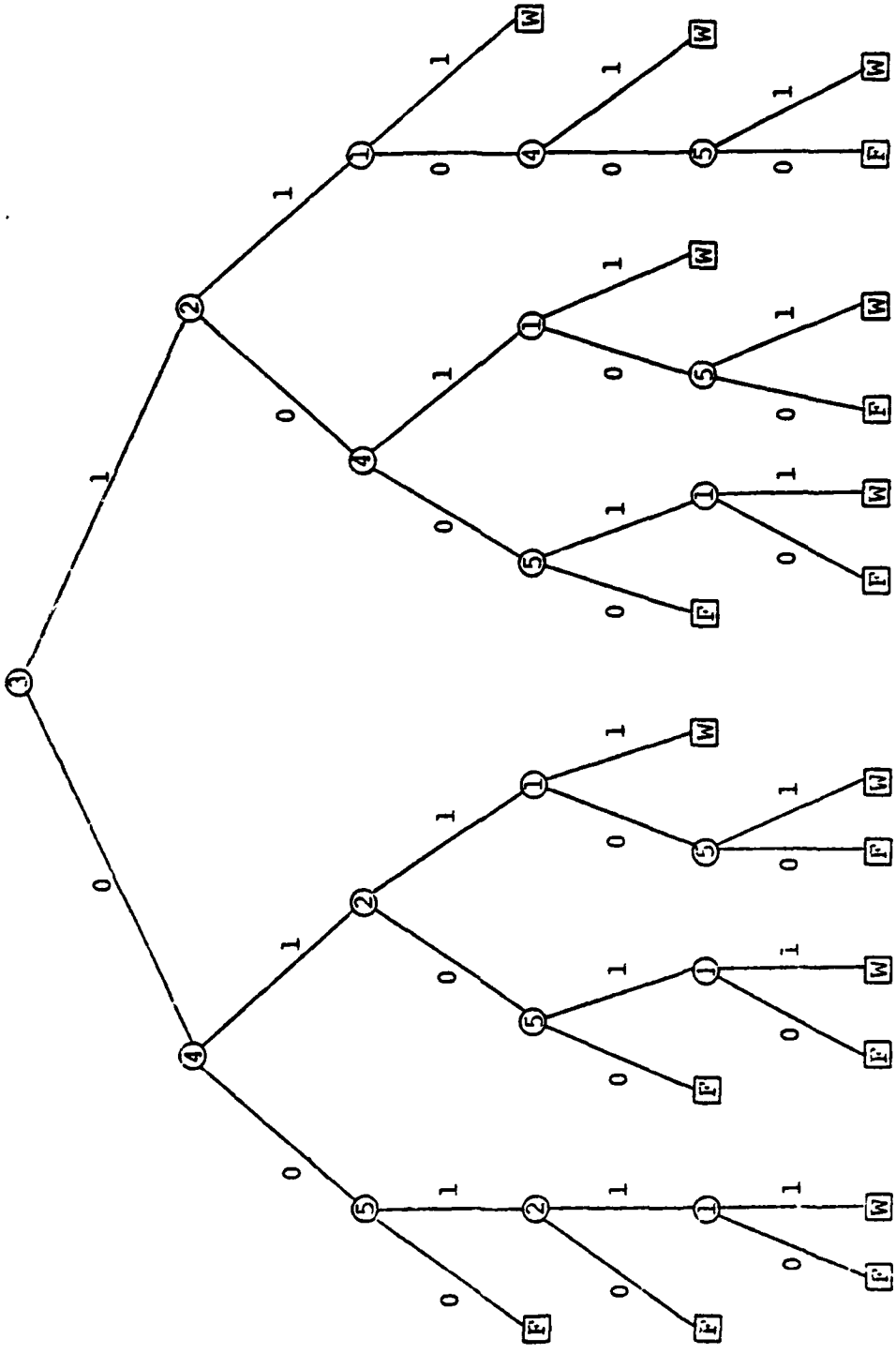


FIGURE 2.1

Proof of Theorem 2.5.1:

First we want to show that knowing the result of the test (i.e., given the information that the system was found to be functioning or failed), and the number of components that were tested is sufficient to identify the tested components according to the given procedure. Knowing the actual components that were tested is sufficient to find the explicit value of the expected cost for the testing procedure which will be calculated later.

Assume that the system is found to be functioning in exactly $k + t$ tests ($t = 0,1, \ldots, n - k$). In such case, exactly t of the tested components will be found to be failed and k will be found to be working, including the last component that was tested (otherwise the procedure would not have stopped with the conclusion that the system is working). In the candidate procedure, the first component to be tested must be among $\{1, \ldots, k\}$, say j. If j is found to be working the next component to be tested would be among the remaining $(k - 1)$ components in the set: $\{\{1, \ldots, k\} - j\}$. If j is found to be failed, the next component to be tested would be among the k components in the expanded set: $\{\{1, \ldots, k,k + 1\} - j\}$. In general, if no failures are found during the testing procedure, there is a need for only k tests, and since we start with components $\{1, \ldots, k\}$ and never expand this set, the tested components will be: $\{1,2, \ldots, k\}$. If only one failure

is found, $k + 1$ tests will be needed to complete the testing of the system, and since the expansion of the initial set was done only once, the tested components will be: $\{1,2, \ldots, k,k + 1\}$. Finally, if the system is found to be working in exactly $k + t$ tests, the set of tested components will be given by: $\{1,2, \ldots, k, k + 1, \ldots, k + t\}$. Thus, exactly t out of the set: $B_t = \{1,2, \ldots, k + t - 1\}$ will be found to be failed, the remaining $k - 1$ components and the $k + t^{th}$ component will be found to be functioning. Denote the probability of the event that exactly m components out of the set B_t are failed and the rest are working by b_m^{k+t-1} .

Consider now the event in which the system is found to be failed in exactly $n - k + t$ tests ($t = 1,2, \ldots, k$). In this event exactly $t - 1$ of the tested components will be found to be functioning, and $n - k + 1$ will be found to be failed including the last component to be tested. Let π be a permutation of the components for which $\frac{c_{\pi_1}}{q_{\pi_1}} \leq \frac{c_{\pi_2}}{q_{\pi_2}} \leq \cdots \leq \frac{c_{\pi_n}}{q_{\pi_n}}$. For the first test, the only candidate components, according to the candidate procedure, are $\{1, \ldots, k\}$. But, for any failure discovered we expand the set by adding a component to it, and since there must be $(n - k + 1)$ failures for the system to fail, eventually *all* the components will be under consideration.

For the first iteration, the tested component must satisfy: $\frac{c_i}{q_i} = \min_{1 \le j \le k} \frac{c_j}{q_j}$, so it must belong to the set: $\{\pi_1, \pi_2, \ldots, \pi_{n-k+1}\}$. If it is found to be failed we expand the set of candidate components for testing, so that the next component to be tested must belong to the set: $\{\pi_1, \pi_2, \ldots, \pi_{n-k}\}$. If the first component is found to be functioning, the next one to be tested must belong to the set: $\{\pi_1, \pi_2, \ldots, \pi_{n-k+2}\}$. In general, if no functioning components are found during the testing procedure, the tested components must be: $\{\pi_1, \pi_2, \ldots, \pi_{n-k+1}\}$. If the system is found to be failed in exactly $n - k + t$ tests, the set of tested components is given by: $\{\pi_1, \pi_2, \ldots, \pi_{n-k+t}\}$. Thus, exactly $t - 1$ out of the set: $A_t = \{\pi_1, \pi_2, \ldots, \pi_{n-k+t-1}\}$ are working, the remaining $n - k$ components and the π_{n-k+t}^{th} component are failed. Denote the probability of the event that exactly m components out of the set A_t are working and the remaining are failed by: $a_m^{n-k+t-1}$.

Now we can construct an induction proof for the theorem, where the induction is made on n , the number of components in the system. For $n = 1$ and $n = 2$ there only exist series and parallel systems.

The candidate procedure reduces to the optimal procedure for these systems, as described in Sections 1 and 2 of this chapter. The same argument holds for any n , when $k = 1$ or $k = n$. Therefore, for the remaining of the proof we can assume that k satisfies: $1 < k < n$.

The induction assumption is that the candidate procedure is optimal for systems with $(n-1)$ components, and we must prove its optimality for systems with n components. First consider the expected cost of testing the system, when we use the given procedure:

$$G^*(k,n) = \underbrace{\sum_{t=0}^{n-k}\left[\sum_{j=1}^{k+t} c_j\right] p_{k+t} b_t^{k+t-1}}_{(*)} + \underbrace{\sum_{t=1}^{k}\left[\sum_{j=1}^{n-k+t} c_{\pi_j}\right] q_{\pi_{n-k+t}} a_{t-1}^{n-k+t-1}}_{(**)} .$$

The expected cost of testing the system is a weighted sum of two conditional costs. By multiplying these costs by the corresponding probabilities of the events we get that (*) is the expected cost when the system is found to be working and (**) is the expected cost when the system is found to be failed.

Recall that the candidate procedure is to test first that component i which satisfies: $\frac{c_i}{q_i} = \min_{1\le j\le k}\left\{\frac{c_j}{q_j}\right\}$. There are two ways in which the first component to be tested can fail to meet the suggested criteria:

(1) When we pick a component i, such that $i > k$.

(2) When the component chosen is $i \le k$, but it does not satisfy the minimum criterion, and hence, according to the above discussion, does not necessarily belong to the set: $(\pi_1, \ldots, \pi_{n-k+1})$, i.e., $i = \pi_\ell$ and $\ell > n - k + 1$.

Since the induction assumption assumes that the procedure is optimal for systems with $(n-1)$ components, it is possible to write the expected cost of testing the system when we start with component i and continue optimally thereafter.

In Case (1) the expected cost is

$$G_i^1(k,n) = \sum_{t=0}^{i-k}\left[c_i + \sum_{j=1}^{k+t-1} c_j\right]p_{k+t-1}\left[q_i b_{t-1}^{k+t-2} + p_i b_t^{k+t-2}\right]$$

$$+ \underbrace{\sum_{t=i-k+1}^{n-k}\left[\sum_{j=1}^{k+t} c_j\right]p_{k+t}b_t^{k+t-1}}_{(*')} + \underbrace{\sum_{t=1}^{k}\left[\sum_{j=1}^{n-k+t} c_{\pi_j}\right]q_{\pi_{n-k+t}}a_{t-1}^{n-k+t-1}}_{(**')}$$

In Case (2), let $i = \pi_\ell$. Then the expected cost is

$$G_i^2(k,n) = \sum_{t=0}^{n-k}\left[\sum_{j=1}^{k+t} c_j\right]p_{k+t}b_t^{k+t-1} + \sum_{t=1}^{\ell-n+k}\left[c_i + \sum_{j=1}^{n-k+t-1} c_{\pi_j}\right]q_{\pi_{n-k+t-1}}$$

$$\left[p_i a_{t-2}^{n-k+t-2} + q_i a_{t-1}^{n-k+t-2}\right] + \sum_{t=\ell-n+k+1}^{k}\left[\sum_{j=1}^{n-k+t} c_{\pi_j}\right]q_{\pi_{n-k+t}}a_{t-1}^{n-k+t-1}.$$

To complete the optimality proof we have to show that: $G^*(k,n) \le G_i(k,n)$ for every i.

The proof is similar for both definitions of $G_i(k,n)$, so the details will be shown only for $G_i^1(k,n)$.

First note that (**) in $G^*(k,n)$ and (**') in $G_i^1(k,n)$ are equal, and also that (*') is equal to the last $(n-i)$ terms of (*). So, we only have to show that for every i:

$$\sum_{t=0}^{i-k}\left[\sum_{j=1}^{k+t} c_j\right]p_{k+t}b_t^{k+t-1} - \sum_{t=0}^{i-k}\left[\left(c_i + \sum_{j=1}^{k+t-1} c_j\right)p_{k+t-1}\left(q_i b_{t-1}^{k+t-2} + p_i b_t^{k+t-2}\right)\right] \le 0$$

but:

$$p_{k+t-1}\left(q_i b_{t-1}^{k+t-2} + p_i b_t^{k+t-2}\right) = p_{k+t-1}\left[(1-p_i)b_{t-1}^{k+t-2} + p_i b_t^{k+t-2}\right]$$

$$= p_{k+t-1}b_{t-1}^{k+t-2} - p_i p_{k+t-1}b_{t-1}^{k+t-2} + p_i p_{k+t-1}b_t^{k+t-2}$$

$$= p_{k+t-1}b_{t-1}^{k+t-2} + p_i q_{k+t-1}b_{t-1}^{k+t-2} - p_i b_{t-1}^{k+t-2} + p_i p_{k+t-1}b_t^{k+t-2}$$

$$= p_i\left(q_{k+t-1}b_{t-1}^{k+t-2} + p_{k+t-1}b_t^{k+t-2}\right) - (p_i - p_{k+t-1})b_{t-1}^{k+t-2}$$

$$= p_i b_t^{k+t-1} - (p_i - p_{k+t-1})b_{t-1}^{k+t-2} ,$$

and hence we need to show that:

$$\sum_{t=0}^{i-k}\left[\sum_{j=1}^{k+t} c_j\right]p_{k+t}b_t^{k+t-1} - \sum_{t=0}^{i-k}\left[\left(c_i + \sum_{j=1}^{k+t-1} c_j\right)\left(p_i b_t^{k+t-1} - p_i b_{t-1}^{k+t-2} + p_{k+t-1}b_{t-1}^{k+t-2}\right)\right] \leq 0 .$$

By convention: $b_{-1}^{k-2} = 0$, and therefore:

$$\sum_{t=0}^{i-k}\left[\sum_{j=1}^{k+t} c_j\right]p_{k+t}b_t^{k+t-1} - \sum_{t=1}^{i-k}\left[\sum_{j=1}^{k+t-1} c_j\right]p_{k+t-1}b_{t-1}^{k+t-2} = \sum_{j=1}^{i} c_j p_i b_{i-k}^{i-1}$$

and also:

$$-\sum_{t=0}^{i-k}\left[c_i + \sum_{j=1}^{k+t-1} c_j\right]p_i b_t^{k+t-1} + \sum_{t=0}^{i-k}\left[c_i + \sum_{j=1}^{k+t-1} c_j\right]p_i b_{t-1}^{k+t-2}$$

$$= \sum_{t=0}^{i-k} c_{k+t}p_i b_t^{k+t-1} - \left[c_i + \sum_{j=1}^{i} c_j\right]p_i b_{i-k}^{i-1} .$$

Finally, we have to show that:

$$\sum_{t=0}^{i-k} [c_{k+t}p_i - p_{k+t}c_i]b_t^{n-k+t} \leq 0$$

but this is true since for any j , $k \leq j \leq i$ we have: $\frac{c_j}{p_j} \leq \frac{c_i}{p_i}$ and the sum from $t = 0$ until $t = i - k$ of nonpositive terms, is nonpositive.

Using the same reasoning for $G_i^2(k,n)$ will result, after the same type of mathematical manipulations, that we have to show that

$$\sum_{t=1}^{\ell-n+k(-1)} \left[c_{\pi_{n-k+t}}q_i - q_{\pi_{n-k+t}}c_i\right] \leq 0 ,$$

but this is true, again, since for any j , $n-k+1 \leq j \leq \ell$ we have: $\frac{c_{\pi_j}}{q_{\pi_j}} \leq \frac{c_i}{q_i}$, and the sum is again nonpositive.

By this the proof for Theorem 2.5.1 is completed.■

3. A BRANCH AND BOUND ALGORITHM FOR MINIMIZING THE EXPECTED COST OF TESTING COHERENT SYSTEMS

3.0 Introduction

Using the concept of Reliability Importance of the components as defined in 1.2.4, we develop a Branch and Bound algorithm to solve the problem of minimizing the expected cost of testing general coherent systems. This algorithm is based on J. D. C. Little's algorithm for solving the travelling salesman problem [18], and it uses the Importance measure as the decision criterion for the branching and for the bounding. In Section 1 we explain the rationale of our algorithm and prove its optimality. In Section 2 we describe the algorithm itself and present a schematic description of its operation. In Section 3 we give two examples of its operation.

3.1 The Reasoning Behind the Algorithm

Recalling from (1.7) the definition for the Reliability Importance of component j : $I_j = \frac{\partial h(\underline{p})}{\partial p_j} = E[\phi(1_j, \underline{X}) - \phi(0_j, \underline{X})]$, let us first define a measure of "unimportance" of component j :

$$\bar{I}_j = 1 - I_j \tag{3.1}$$

and investigate its meaning. Suppose we are given two structural vectors: $\underline{X}^1$ and $\underline{X}^2$ such that:

(i) $\phi(\underline{X}^1) = \phi(\underline{X}^2)$ and (ii) $x_i^1 = x_1^2$ for every $i \neq i_o$, but $x_{i_o}^1 \neq x_{i_o}^2$. We want to find the value of $\phi(\underline{X}^1)$ (or, $\phi(\underline{X}^2)$), and to do so by testing the individual components. One way of doing this would be to test all the components in an arbitrary order and thus find the value of $\phi(\underline{X}^1)$. This would definitely be unnecessarily costly, since the value of $\phi(\underline{X}^1)$ is independent of the value of x_{i_o}, and we therefore need not check this component.

The measure of unimportance which we defined can be described as follows: we find the total number of vectors for which the checking of a specific component is unnecessary, and then look at the probability of the occurrence of those vectors, i.e.,

$$\text{Prob}\{j \text{ is unimportant}\} = \bar{I}_j =$$

$$1 - I_j = P\{[\phi(1_j, \underline{X}) - \phi(0_j, \underline{X})] = 0\} .$$

Next we define

$$d_j = c_j \bar{I}_j \tag{3.2}$$

which is a measure of the expected extra cost due to testing component j even though it is unnecessary to know its state. Thus: $e_j = c_j I_j$ is a measure of the expected necessary expense due to testing component j. This is equal to the cost of testing component j minus the

expected cost of the extra expense, i.e.,

$$e_j = c_j I_j = c_j - d_j \ .$$

We define the total expected necessary expense to be:

$$(3.3) \quad I = \sum_k c_k I_k = \sum_k c_k (1 - \bar{I}_k) = \sum_k c_k - \sum_k c_k \bar{I}_k = \sum_k c_k - \sum_k d_k \ .$$

Finally, we introduce some additional notation:

$$(3.4) \quad I_j(i) = h(1_j, 1_i, \underline{P}) - h(0_j, 1_i, \underline{P})$$

$$(3.5) \quad I_j(i') = h(1_j, 0_i, \underline{P}) - h(0_j, 0_i, \underline{P}) \ ,$$

where $I_j(i)$ and $I_j(i')$ are the measures of the importance of component j, when component i is given to be functioning or failed respectively.

Also, let:

$$(3.6) \quad d_j(i) = c_j \bar{I}_j(i)$$

$$(3.7) \quad d_j(i') = c_j \bar{I}_j(i') \ .$$

In general, we define:

$$(3.8) \quad I_j(\underline{Z}, s) = h(1_j, \underline{Z}, 1_s, \underline{P}) - h(0_j, \underline{Z}, 1_s, \underline{P})$$

$$(3.9) \quad I_j(\underline{Z}, s') = h(1_j, \underline{Z}, 0_s, \underline{P}) - h(0_j, \underline{Z}, 0_s, \underline{P})$$

where $I_j(\underline{Z}, s)$ and $I_j(\underline{Z}, s')$ are the measures of the importance of component j, given the vector of known

component states $\underline{Z}$, and component s is given to be functioning or failed respectively. Also, let:

$$d_j(\underline{Z},s) = c_j\bar{I}_j(\underline{Z},s) \tag{3.10}$$

and

$$d_j(\underline{Z},s') = c_j\bar{I}_j(\underline{Z},s') \ . \tag{3.11}$$

Following the above discussion, one can suggest the following algorithm: Start the testing procedure by checking component i , which minimizes d_k for $k = 1,2, \ldots, n$. Then continue to develop the testing tree by choosing the components that minimize $d_j(i)$ and $d_k(i')$ for $k , j = 1,2, \ldots, n ; k , j \neq i$, and continue in this manner. This method, however, does not necessarily give an optimal testing procedure. Actually, we can show by induction (see Theorem 3.1.2) that $I + d_i$ is a lower bound for the expected cost of all testing procedures which start by testing component i . As a corollary, it is easy to see that $I + \min_i \{d_i\}$ is the minimum expected cost of *all* testing procedures. Therefore, we must modify the "naive" algorithm suggested above in order to be able to consider testing procedures which have a higher lower bound, but yet may give a smaller expected cost.

Before proving the first theorem, we present a lemma which states that testing component i first does not change the expected extra cost of testing any of the other components.

3.1.1 Lemma

$$\sum_{k\neq i} d_k = p_i \sum_k d_k(i) + (1 - p_i) \sum_k d_k(i') \ .$$

Proof:

$$(3.12) \quad \sum_{k\neq i} d_k = \sum_{k\neq i} c_k \bar{I}_k = \sum_{k\neq i} c_k\{1 - [h(1_k,\underline{P}) - h(0_k,\underline{P})]\} \ .$$

$$\sum_k d_k(i) = \sum_{k\neq i} c_k \bar{I}_k(i) = \sum_{k\neq i} c_k\{1 - [h(1_k,1_i,\underline{P}) - h(0_k,1_i,\underline{P})]\}$$

$$\sum_k d_k(i') = \sum_{k\neq i} c_k \bar{I}_k(i') = \sum_{k\neq i} c_k\{1-[h(1_k,0_i,\underline{P}) - h(0_k,0_i,\underline{P})]\}$$

$$\therefore p_i \sum_k d_k(i) + (1 - p_i) \sum_k d_k(i')$$

$$= \sum_{k\neq i} c_k - \sum_{k\neq i} c_k\{p_i[h(1_k,1_i,\underline{P}) - h(0_k,1_i,\underline{P})]$$

$$+ (1 - p_i)[h(1_k,0_i,\underline{P}) - h(0_k,0_i,\underline{P})]\}$$

$$= \sum_{k\neq i} c_k - \sum_{k\neq i} c_k \{[p_i h(1_k,1_i,\underline{P}) + (1 - p_i)h(1_k,0_i,\underline{P})]$$

$$(3.13) \qquad - [p_i h(0_k,1_i,\underline{P}) + (1 - p_i)h(0_k,0_i,\underline{P})]\}$$

but, recalling that:

$$h(\underline{P}) = p_i h(1_i,\underline{P}) + (1 - p_i)h(0_i,\underline{P})$$

we also have:

$$h(1_k,\underline{P}) = p_i h(1_k,1_i,\underline{P}) - (1 - p_i)h(1_k,0_i,\underline{P})$$

and:

$$h(0_k,\underline{P}) = p_i h(0_k,1_i,\underline{P}) - (1 - p_i)h(0_k,0_i,\underline{P})$$

$$\therefore (3.13) = \sum_{k\neq i} c_k - \sum_{k\neq i} c_k[h(1_k,\underline{P}) - h(0_k,\underline{P})]$$

$$= \sum_{k\neq i} c_k\{1 - [h(1_k,\underline{P}) - h(0_k,\underline{P})]\} = (3.12) \ . \blacksquare$$

3.1.2 Theorem

A lower bound for the expected cost of all testing procedures which start by testing component i, is: $I + d_i$.

Proof:

We use an induction proof, where the induction is based on n, the number of components in the system.

For $n = 1$ the proof is trivial. For $n = 2$ there are two possible systems with 2 components: a series system and a parallel system. We study them in turn.

(i) *Series System*: $h(\underline{P}) = p_1 p_2$.

$$d_1 = c_1(1 - I_1) = c_1\{1 - [h(1_1,p_2) - h(0_1,p_2)]\} =$$

$$c_1(1 - p_2 + 0) = c_1 q_2 \ .$$

$$d_2 = c_2(1 - I_2) = c_2\{1 - [h(p_1,1_2) - h(p_1,0_2)]\} =$$

$$c_2(1 - p_1 + 0) = c_2 q_1 \ .$$

$$\therefore I = \sum_{i=1}^{2} c_i - \sum_{i=1}^{2} d_i = c_1 + c_2 - c_1 q_2 - c_2 q_1 =$$

$$c_1 p_2 + c_2 p_1 \ .$$

Finally:

$$I + d_1 = c_1p_2 + c_2p_1 + c_1q_2 = c_1 + p_1c_2$$

$$I + d_2 = c_1p_2 + c_2p_1 + c_2q_1 = c_2 + p_2c_1 \ .$$

Those lower bounds are actually equal to the expected cost of testing the system, when we start by checking component 1 or 2 respectively.

(ii) *Parallel System*: $h(\underline{P}) = p_1 + p_2 - p_1p_2$.

$$d_1 = c_1(1 - I_1) = c_1\{1 - [h(1_1,p_2) - h(0_1,p_2)]\} =$$

$$c_1(1 - 1 + p_2) = c_1p_2 \ .$$

$$d_2 = c_2(1 - I_2) = c_2\{1 - [h(p_1,1_2) - h(p_1,0_2)]\} =$$

$$c_2(1 - 1 + p_1) = c_2p_1 \ .$$

$$\therefore I = \sum_{i=1}^{2} c_i - \sum_{i=1}^{2} d_i = c_1 + c_2 - c_1p_2 - c_2p_1 =$$

$$c_1q_2 + c_2q_1 \ .$$

Therefore:

$$I + d_1 = c_1q_2 + c_2q_1 + c_1p_2 = c_1 + q_1c_2$$

$$I + d_2 = c_1q_2 + c_2q_1 + c_2p_1 = c_2 + q_2c_1 \ .$$

Here, again, the lower bounds are equal to the expected costs.

Now we assume that the theorem is true for systems with $(n-1)$ components, and show it is true for systems with n components. Assume that the first component to be tested is i, and let C_i be the optimal expected cost of testing the system, if we start the testing procedure by checking component i first. i can be found to be functioning or failed and we denote by $C(i)$ and $C(i')$ the optimal costs of testing the subtrees that can be developed respectively. Then we have the equation:

$$C_i = c_i + p_i C(i) + q_i C(i') \quad . \tag{3.15}$$

We want to show that: $C_i \geq I + d_i$ for every $i = 1, 2, \ldots, n$. By the induction hypothesis we have:

$$C(i) \geq I(i) = \sum_{k \neq i} c_k - \sum_k d_k(i)$$

$$C(i') \geq I(i') = \sum_{k \neq i} c_k - \sum_k d_k(i')$$

$$\therefore C_i \geq c_i + p_i\left[\sum_{k \neq i} c_k - \sum_k d_k(i)\right] + q_i\left[\sum_{k \neq i} c_k - \sum_k d_k(i')\right]$$

$$= \sum_{k \neq i} c_k - \left[p_i \sum_k d_k(i) + q_i \sum_k d_k(i')\right]$$

$$= \sum_k c_k - \sum_{k \neq i} d_k \quad \text{(by the lemma)}$$

$$= \sum_k c_k - \sum_k d_k + d_i$$

$$= I + d_i \quad . \blacksquare$$

Corollary:

A lower bound for the expected cost of all the testing procedures is given by: $I + \min_i \{d_i\}$.

The next theorem allows us to speed up the calculations of the lower bounds in each iteration of our algorithm.

3.1.3 Theorem

A lower bound for the expected cost of testing the system - given that we start the testing procedure by checking component i , and then testing components α or β depending on whether $x_i = 1$ or $x_i = 0$ respectively - is given by:

$$L_i = I + d_i + p_i d_\alpha(i) + q_i d_\beta(i') \ .$$

Proof:

$$\begin{aligned} L_i &= c_i + p_i \left[\sum_{k \neq i} c_k - \sum_k d_k(i) + d_\alpha(i) \right] \\ &\quad + q_i \left[\sum_{k \neq i} \tau_k - \sum_k d_k(i') + d_\beta(i') \right] \\ &= \sum_k c_k - p_i \sum_k d_k(i) + q_i \sum_k d_k(i') \\ &\quad + p_i d_\alpha(i) + q_i d_\beta(i') \\ &= \sum_k c_k - \sum_{k \neq i} d_k + p_i d_\alpha(i) + q_i d_\beta(i') \\ &= I + d_i + p_i d_\alpha(i) + q_i d_\beta(i') \ . \blacksquare \end{aligned}$$

A branch and bound algorithm seems to be suitable for our problem, and we use the measure of unimportance that we defined in (3.1) as the decision criterion for its operation. The algorithm is described in the next section.

3.2 Branch and Bound Algorithm

To establish the optimal testing procedure we need to develop an optimal tree that will tell us which component to test first, and according to the result of the first test, or any other subsequent tests where to continue in our testing procedure. The algorithm works by developing subtrees, such that at each iteration the subtree on hand is the one which gives the lowest bound of the type that we defined earlier.

We start the algorithm with n such subtrees, which are the original components, and pick the one which minimizes $I + d_i$ over all $i = 1,2, \ldots, n$. With this component k, say, we continue by considering the $(n - 1)^2$ subtrees which consist of the component k branching to each combination of two out of the $(n - 1)$ remaining components. For each of these subtrees, which correspond to different possible decision rules, the left hand branch goes to the component which is to be tested if k is found failed, and the right hand branch goes to the component which is to be tested if k is found working.

After computing the bounds for each of these new subtrees, we find the minimum of all the bounds that we have computed so far, and continue to expand the corresponding subtree in a similar manner. This subtree may, at this stage, be either a single component, or a subtree which we have constructed.

Eventually we will find a subtree which is a complete tree (i.e., it need not be expanded any more, and thus it uniquely describes a testing procedure for the given system) such that its expected cost is less than or equal to the lower bound for any other subtree. This will be the optimal testing tree.

The algorithm is described schematically in Figure 3.1.

3.3 Examples

i) Consider the 2-out-of-3 system with the reliability function: $h(\underline{p}) = p_1p_2 + p_1p_3 + p_2p_3 - {}^2p_1p_2p_3$, and with the following data:

i	p_i	c_i	c_i/p_i	c_i/q_i
1	.6	5	8.3	1.25
2	.8	2	2.5	10
3	.9	4	4.4	40

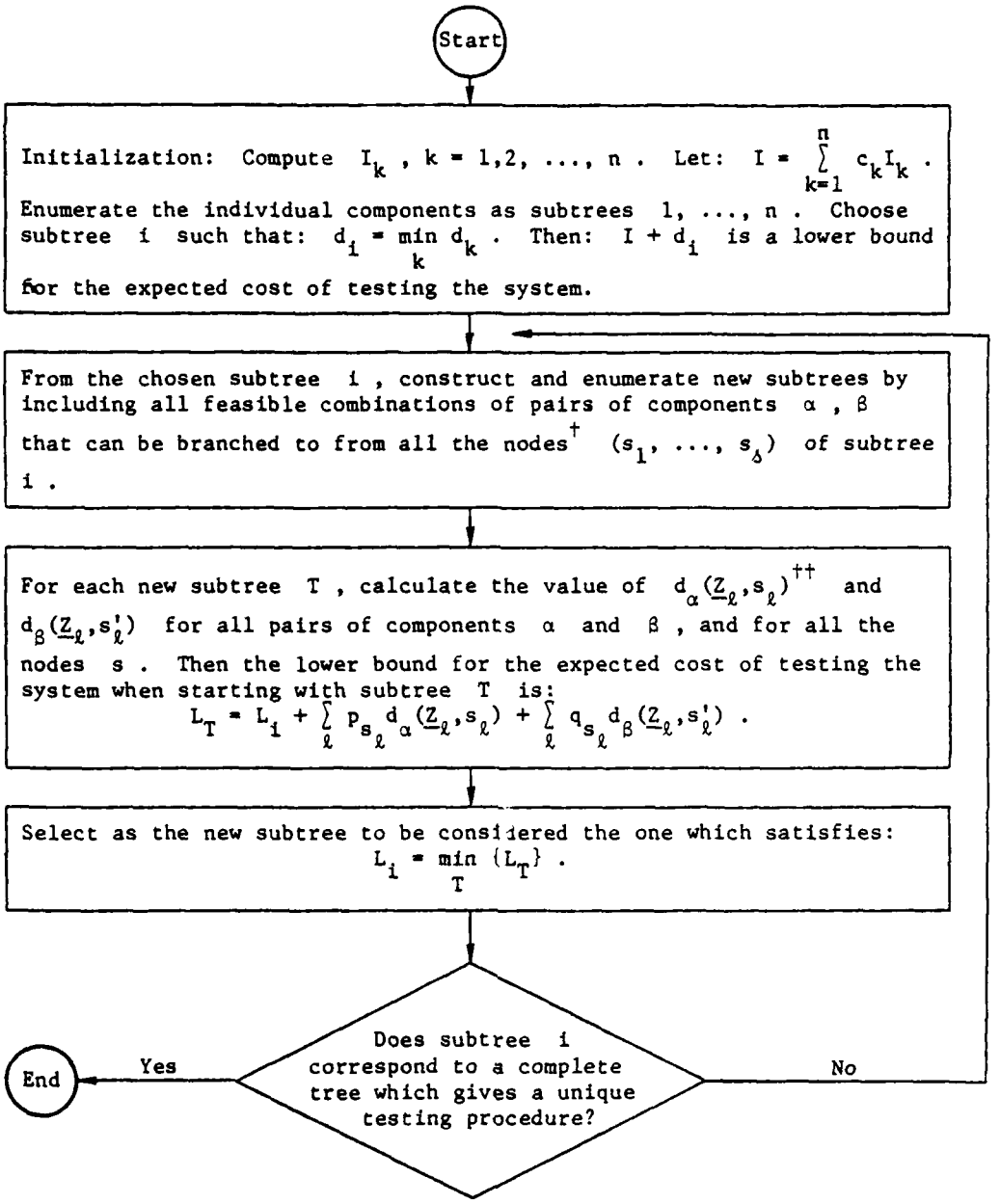


FIGURE 3.1

---

†Nodes are the lowest level components in a subtree.

††$\underline{Z}_\ell$ is the vector of component states which lead to node s_ℓ in subtree i .

Initialization:

Compute the importance measures for all the components.

$$I_1 = p_2 + p_3 - 2p_2p_3 = 1.7 - 1.44 = .26$$

$$I_2 = p_1 + p_3 - 2p_1p_3 = 1.5 - 1.08 = .42$$

$$I_3 = p_1 + p_2 - 2p_1p_2 = 1.4 - .96 = .44$$

$$I = \sum_{i=1}^{3} c_i I_i = 3.9 \ .$$

Iteration #1:

Compute the lower bounds as defined in Theorem 3.1.2 for all the given components: (1) ① (2) ② (3) ③

$$L_1 = I + d_1 = I + c_1(1 - I_1) = 3.9 + 3.7 = 7.6$$

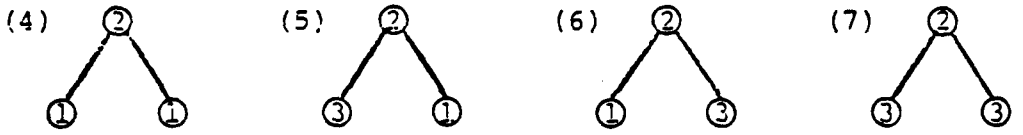
$$L_2 = I + d_2 = I + c_2(1 - I_2) = 3.9 + 1.16 = 5.06$$

$$L_3 = I + d_3 = I + c_3(1 - I_3) = 3.9 + 2.24 = 6.14 \ .$$

The minimum is L_2 , and we continue by developing a subtree which its root is component 2 .

Iteration #2:

Compute the lower bounds as defined in Theorem 3.1.3 for the following trees:



Actually, we first have to compute:

$$d_3(2) = c_3(1 - 1 + p_1) = p_1c_3 = 2.4$$

$$d_3(2') = c_3(1 - p_1) = q_1c_3 = 1.6$$

$$d_1(2) = c_1(1 - 1 + p_3) = p_3c_1 = 4.5$$

$$d_1(2') = c_1(1 - p_3) = q_3c_1 = .5 .$$

The new set of lower bounds will be:

$$L_4 = L_2 + p_2d_1(2) + q_2d_1(2') = 5.06 + 3.7 = 8.76$$

$$L_5 = L_2 + p_2d_1(2) + q_2d_3(2') = 5.06 + 3.92 = 8.98$$

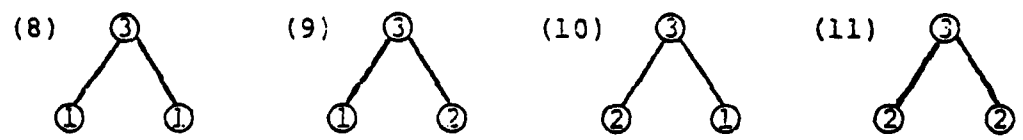
$$L_6 = L_2 + p_2d_3(2) + q_2d_1(2') = 5.06 + 2.02 = 7.08$$

$$L_7 = L_2 + p_2d_3(2) + q_2d_3(2') = 5.06 + 2.24 = 7.3 .$$

Now since $L_6 > L_3$, we have to go back and develop the trees that can be originated in L_3 , since at this stage $L_3 = \min_{\substack{i=1,\ldots,7 \\ i\neq 2}} L_i$.

Iteration #3 (Termination):

Compute the lower bounds for the following trees:



First we compute the new $g_i(3)$ and $g_i(3')$:

$$d_1(3) = c_1(1 - 1 + p_2) = c_1p_2 = 4$$

$$d_1(3') = c_1(1 - p_2) = c_1q_2 = 2$$

$$d_2(3) = c_2(1 - 1 + p_1) = c_2p_1 = 1.2$$

$$d_2(3') = c_2(1 - p_1) = c_2q_1 = 0.8 .$$

The new set of lower bounds will be:

$$L_8 = L_3 + p_3d_1(3) + q_3d_1(3') = 6.14 + 3.6 + .2 = 9.94$$

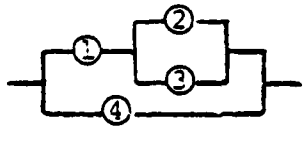
$$L_9 = L_3 + p_3d_2(3) + q_3d_1(3') = 6.14 + 1.08 + .2 = 7.42$$

$$L_{10} = L_3 + p_3d_1(3) + q_3d_2(3') = 6.14 + 3.6 + .08 = 9.82$$

$$L_{11} = L_3 + p_3d_2(3) + q_1d_2(3') = 6.14 + 1.08 + .08 = 7.3 .$$

Now, $L_6 = \min_{\substack{i=1,\ldots,11 \\ i\neq 2,3}} \{L_i\}$, and note that the tree given in (6) is a complete tree for our problem. The complete testing procedure can be described by an equivalent tree (see Section 2.5), and the expected cost of testing the system is equal in both cases.

ii) Consider the following system:



where: $h(\underline{p}) = p_4 + p_1p_2 + p_1p_3 - p_1p_2p_3 - p_1p_2p_4 - p_1p_3p_4 + p_1p_2p_3p_4$, and with the

following data:

i	p_i	c_i	c_i/p_i	c_i/q_i
1	.5	10	20	20
2	.5	1	2	2
3	.5	1000	2000	2000
4	.5	100	200	200

Initialization:

Compute the importance measures for the components:

$$I_1 = p_2 + p_3 - p_2p_3 - p_2p_4 - p_3p_4 + p_2p_3p_4 = .375$$

$$I_2 = p_1 - p_1p_3 - p_1p_4 + p_1p_3p_4 = .125$$

$$I_3 = p_1 - p_1p_2 - p_1p_4 + p_1p_2p_4 = .125$$

$$I_4 = 1 - p_1p_2 - p_1p_3 + p_1p_2p_3 = .625$$

$$I = \sum_{i=1}^{4} c_i I_i = 191.375 .$$

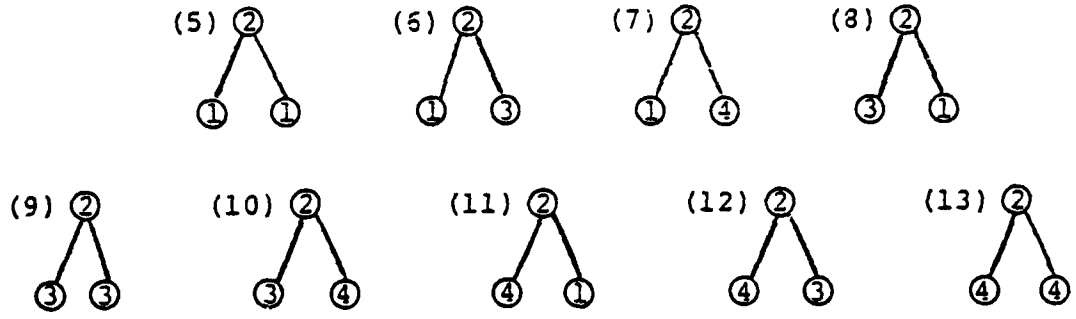
Iteration #1:

Compute the lower bounds for: (1) ① (2) ② (3) ③ (4) ④ :

$$\left.\begin{aligned} L_1 &= I + d_1 = 191.375 + 6.25 = 197.625 \\ L_2 &= I + d_2 = 191.375 + .875 = 192.25 \\ L_3 &= I + d_3 = 191.375 + 875 = 1066.375 \\ L_4 &= I + d_4 = 191.375 + 37.5 = 228.875 \end{aligned}\right\} \Rightarrow \min_i \{L_i\} = L_2 .$$

Iteration #2:

Compute the lower bounds for the following subtrees:



$$d_1(2) = c_1(1 - 1 + p_4) = 2.5$$

$$d_1(2') = c_1(1 - p_3 + p_3p_4) = 7.5$$

$$d_3(2) = c_1(1 - 0) = 1000$$

$$d_3(2') = c_3(1 - p_1 + p_1p_4) = 750$$

$$d_4(2) = c_4(1 - 1 + p_1) = 50$$

$$d_4(2') = c_4(1 - 1 + p_1p_3) = 25 .$$

The new set of lower bounds will be:

$$L_5 = L_2 + p_2d_1(2) + q_2d_1(2') = 5 + 192.25 = 197.25$$

$$L_6 = L_2 + p_2d_3(2) + q_2d_1(2') = 503.75 + 192.25 = 696$$

$$L_7 = L_2 + p_2d_4(2) + q_2d_1(2') = 28.75 + 192.25 = 221$$

$$L_8 = L_2 + p_2d_1(2) + q_2d_3(2') = 376.25 + 122.25 = 568.5$$

$$L_9 = L_2 + p_2d_3(2) + q_2d_3(2') = 875 + 192.25 = 1067.25$$

$$L_{10} = L_2 + p_2d_4(2) + q_2d_3(2') = 400 + 192.25 = 592.25$$

$$L_{11} = L_2 + p_2 d_1(2) + q_2 d_4(2') = 13.75 + 192.25 = 206$$

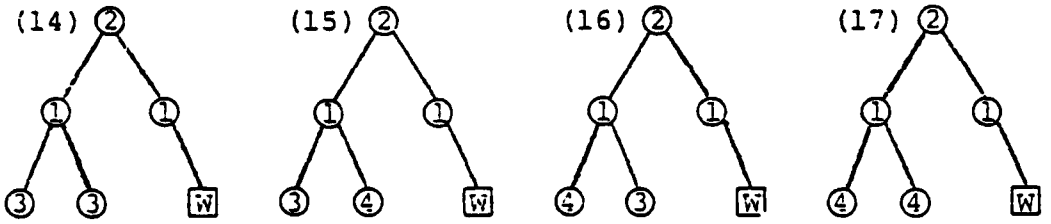
$$L_{12} = L_2 + p_2 d_3(2) + q_2 d_4(2') = 512.5 + 192.25 = 704.75$$

$$L_{13} = L_2 + p_2 d_4(2) + q_2 d_4(2') = 37.5 + 192.25 = 229.75 .$$

$L_5 = \min_{\substack{i=1,\ldots,13 \\ i \neq 2}} \{L_i\}$, so we continue by developing this subtree.

Iteration #3:

Note that if both components 2 and 1 are working - the system is working, so we only have to develop the left side of our subtree:



$$d_3(2',1) = c_3(1 - 1 + p_4) = 250$$

$$d_3(2',1') = c_3(1 - 0) = 1000$$

$$d_4(2',1) = c_4(1 - 1 + p_3) = 25$$

$$d_4(2',1') = c_4(1 - 1) = 0 .$$

$$L_{14} = L_5 + p_1 d_3(2',1) + q_1 d_3(2',1') = 197.25 + 625 = 822.25$$

$$L_{15} = L_5 + p_1 d_4(2',1) + q_1 d_3(2',1') = 197.25 + 512.5 = 709.75$$

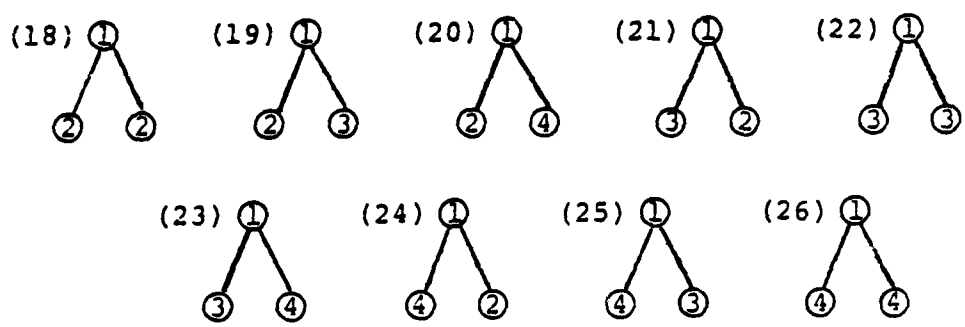
$$L_{16} = L_5 + p_1 d_3(2',1) + q_1 d_4(2',1') = 197.25 + 125 = 322.25$$

$$L_{17} = L_5 + p_1 d_4(2',1) + q_1 d_4(2',1') = 197.25 + 12.5 = 209.75 .$$

$L_1 = \min_{\substack{i=1,\ldots,17 \\ i\neq 2,5}} \{L_i\}$, and we have to go back to develop a new first level tree.

Iteration #4:

Compute the lower bounds for the following subtrees:



$$d_2(1) = c_2(1 - 1 + p_3 + p_4 - p_3p_4) = .75$$

$$d_2(1') = c_2(1 - 0 = 1$$

$$d_3(1) = c_3(1 - 1 + p_2 + p_4 - p_2p_4) = 750$$

$$d_3(1') = c_3(1 - 0) = 1000$$

$$d_4(1) = c_4(1 - 1 + p_2 + p_3 - p_2p_3) = 75$$

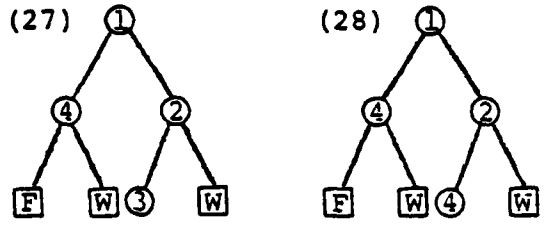
$$d_4(1') = c_4(1 - 1) = 0 .$$

Using the above rules it is easy to verify that

$$L_{24} = \min_{\substack{i=1,\ldots,26 \\ i\neq 2,5}} \{L_i\} = 197.625 + .375 = 198 .$$

Iteration #5:

Compute the lower bounds for the following subtrees:

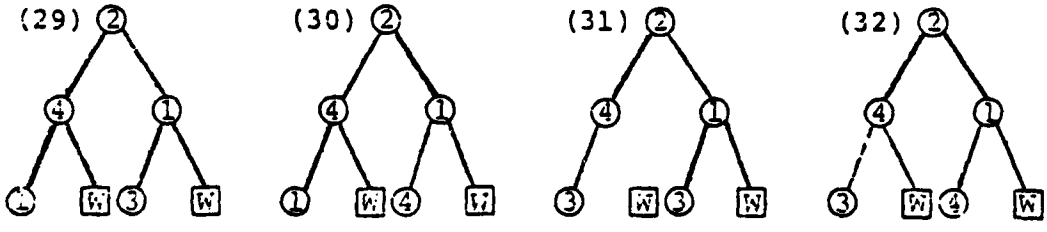


$$d_3(1,2') = c_3(1 - 1 + p_4) = 250$$

$$d_4(1,2') = c_4(1 - 1 + p_3) = 25 .$$

$$\left.\begin{aligned} \therefore\ L_{27} &= L_{24} + q_2 d_3(1,2') = 198 + 125 = 323 \\ L_{28} &= L_{24} + q_2 d_4(1,2') = 198 + 12.5 = 210.5 \end{aligned}\right\} \Rightarrow L_{11} = \min_{\substack{i=1,\ldots,28 \\ i\neq 1,2,5,24}} \{L_i\} .$$

Iteration #6 (Termination):



$$d_1(2',4') = c_1(1 - p_3) = 5$$

$$d_3(2',4') = c_3(1 - p_1) = 500$$

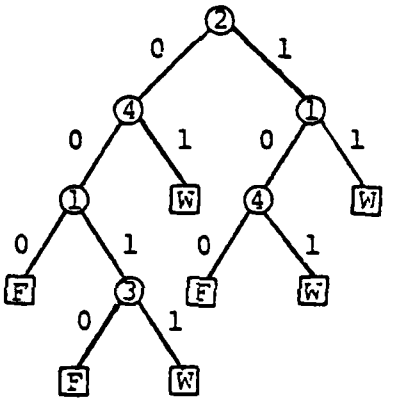
$$d_3(2,1') = c_3(1 - 0) = 1000$$

$$d_4(2,1') = c_4(1 - 1) = 0 .$$

$$L_{30} = L_{11} + q_4 d_1(2',4') + q_1 d_4(2,1') = 206 + .5.5 + .5.0 = 208.5 \ .$$

Now it is easy to show that: $L_{30} = \min_{\substack{i=1,\ldots,32 \\ i \neq 1,2,5,11,24}} \{L_i\} =$ 208.5 . Also, since this is a complete tree which uniquely describes a testing procedure, we finally get our optimal solution.

An equivalent representation of the optimal search tree is:



The expected cost of testing is given by:

$$C = c_2 + (p_2 + q_2q_4)c_1 + (q_2 + p_2q_1)c_4 + q_2q_4p_1c_3$$

$$= 1 + .75 \cdot 10 + .75 \cdot 100 + .125 \cdot 1000 = 208.5 \ .$$

4. ON THE SOLUTION OF SOME RELATED PROBLEMS

4.0 Introduction

Some unsolved related problems are presented in this chapter, together with the attempts that were made towards finding a solution. For the most general type of coherent systems we do not believe that an *efficient* algorithm can be found, which will solve the problem of minimizing the expected cost of testing a system, for any given system. However, for the special case of coherent systems which can be represented as two terminal networks without replicated components, a procedure which is more efficient than the branch and bound algorithm of the previous chapter, may exist. In Section 1 we introduce a set of rules which can be applied to any general coherent system, and we believe that its usage is efficient in particular for the cases of two terminal networks. In Section 2 we use the idea of modularization (introduced in Chapter 1) to get an application for fault tree analysis. Modules seem to be very useful in some aspects of reliability theory and fault tree analysis (e.g., getting tighter bounds for the reliability of a complex network), but they do not seem to help us much in our context. Finally, in Section 3, we introduce a closely related problem, that of how to minimize the expected cost of finding the failed components - given that the system has failed, and give solutions for some special cases of this problem.

4.1 Minimizing the Expected Cost of Testing General Coherent Systems

In Chapter 2 we gave efficient rules for solving the problem for some special cases. In Chapter 3 we introduced a branch-and-bound algorithm which solves the problem for any general system. Unfortunately, branch-and-bound algorithms are in general very slow and inefficient, and sometimes are even impossible to perform for large systems. Therefore it is necessary to look for some sub-optimal rules which are easier to compute, but usually give only sub-optimal procedures for the testing problem.

Recall from 1.1.7 that every coherent system can be represented either in terms of minimal-paths or minimal-cuts. Those representations are similar to the parallel-series and series-parallel structures respectively, except that in the general case they do not satisfy the requirement of disjoint minimal path-sets or disjoint minimal cut-sets respectively. Nevertheless, we suggest the use of the optimal rules for parallel-series and series-parallel systems on the minimal-path and minimal-cut representation. Applying the algorithm for parallel-series systems will result in getting a component, i, which is a candidate for being the first component to be tested under our sub-optimal procedure. Applying the algorithm for series-parallel systems on the minimal cut representation will result in getting a component, j, which is also a candidate for being the

first component to be tested. We now face two different situations:

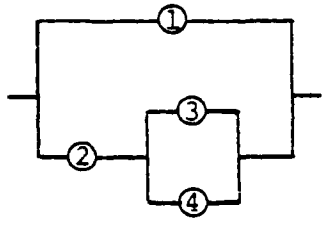
(i) $i = j$. If, by applying the two algorithms to the two different representations of the coherent system we get the same component as a candidate for the first test, we should actually test this component first. After testing this component we have a new system with $(n - 1)$ components. We find its minimal-path and minimal-cut representation and apply the parallel-series and the series-parallel algorithms to the above representations respectively, and continue in this manner.

(ii) $i \neq j$. In this case we have two candidates for the first test. This means that if we want to develop the tree which represents the testing procedure, we have two possible roots for the tree. This is still much better than having n possible roots, and we think that one of these trees, which uses either component i or component j as a root, is the optimal tree for the testing procedure. In this case, as above, after testing the i^{th} or the j^{th} component, we have a new system with only $(n - 1)$ components. Applying the same rules again may result in

having 4 trees as candidates for optimal testing procedure, and the number will increase in every iteration. Nevertheless using this procedure is certainly better than the regular dynamic programming approach which was introduced in the preface, and it may even be better than the branch and bound algorithm which is described in the previous chapter.

Example:

Consider the following coherent system:



i	c_i	p_i
1	30	.5
2	15	.25
3	12	.6
4	10	.5

with the structure function:

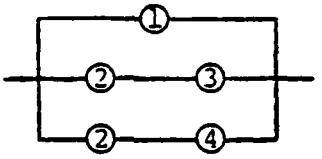
$$\phi(\underline{X}) = 1 - (1 - x_1)\{1 - x_2[1 - (1 - x_3)(1 - x_4)]\} .$$

The minimal path-sets and minimal cut-sets of the system are given by:

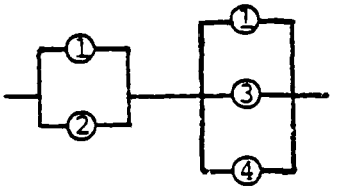
$$R_1 = \{1\}$$
$$R_2 = \{2,3\}$$
$$R_3 = \{2,4\}$$

$$K_1 = \{1,2\}$$
$$K_2 = \{1,3,4\} .$$

Therefore, the minimal-path representation of the system is given by:



and the minimal-cut representation is:



Note that: $R_2 \cap R_3 = 2 \neq \phi$ and: $K_1 \cap K_2 = 1 \neq \phi$, so the system is not a series-parallel nor it is a parallel-series structure. Using (2.3), (2.4), (2.5), (2.6) we can calculate:

$E(R_1) = 30$ $\qquad$ $P(R_1) = .5$

$E(R_2) = 15 + 3 = 18$ $\qquad$ $P(R_2) = .25 \cdot .6 = .15$

$E(R_3) = 15 + 2.5 = 17.5$ $\qquad$ $P(R_3) = .25 \cdot .5 = .125$

$$\therefore \frac{E(R_1)}{P(R_1)} = \frac{30}{.5} = 60 \; ; \; \frac{E(R_2)}{P(R_2)} = \frac{18}{.15} = 120 \; ; \; \frac{E(R_3)}{P(R_3)} = \frac{17.5}{.125} = 140 \; .$$

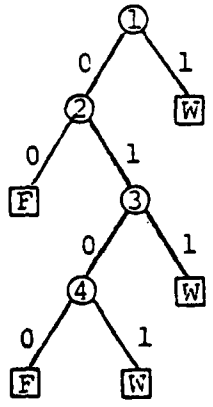
Also:

$F(K_1) = 30 + 7.5 = 37.5$ $\qquad$ $Q(K_1) = .375$

$F(K_2) = 10 + 6 + 6 = 22$ $\qquad$ $Q(K_2) = .1$

$$\therefore \frac{F(K_1)}{Q(K_1)} = \frac{37.5}{.375} = 100 \; ; \; \frac{F(K_2)}{Q(K_2)} = \frac{22}{.1} = 220 \; .$$

Since $\frac{E(R_1)}{P(R_1)} = \min_{j=1,2,3} \left\{ \frac{E(R_j)}{P(R_j)} \right\}$ and $\frac{F(K_1)}{Q(K_1)} = \min_{j=1,2} \left\{ \frac{F(K_j)}{Q(K_j)} \right\}$, and since component number 1 is the first component to be tested in this path set and in this cut set respectively, we conclude that component 1 should be the first one to be tested under the optimal testing procedure. After testing component 1 we are left with a series-parallel system, and we continue this example using the rule in Section 2.4. The optimal search tree for this problem will then be:



and its expected cost is given by:

$$C = c_1 + q_1 c_2 + q_1 p_2 c_3 + q_1 p_2 q_3 c_4$$

$$= 30 + .5 \cdot 15 + .125 \cdot 12 + .05 \cdot 10$$

$$= 30 + 7.5 + 1.5 + .5 = 39.5 \; .$$

Using the branch and bound algorithm it is easy to verify that this is actually the optimal tree for testing this system.

4.2 Application to Fault Tree Analysis

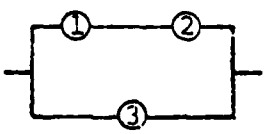
Fault Tree Analysis (FTA) is a methodology for evaluating the failure processes of complex systems. Whereas Reliability Theory is mainly concerned with the ability of a system to perform its desired mission, the Fault Tree Analysis technique investigates the undesired and hazardous events. In Chapter 1 we specified the characteristics of coherent systems. An alternative representation of coherent systems are fault trees. Both representations have the common feature of "basic" events, the component states of the coherent systems, for which we are given the probabilities of being functioning or failed. Using either the structure function, or the structure of the tree, we can evaluate the probability of the occurrence of the "top" event. In Reliability Theory this event is the proper functioning of the system. In Fault Tree Analysis the top event is the improper functioning of the system which would lead to a hazardous condition.

Given a complex fault tree, it is not an easy task to get its equivalent representation as a coherent structure. FTA has become a standard and widely used technique for evaluating the operation of complex systems in fields such as nuclear and chemical engineering, electrical engineering, etc.

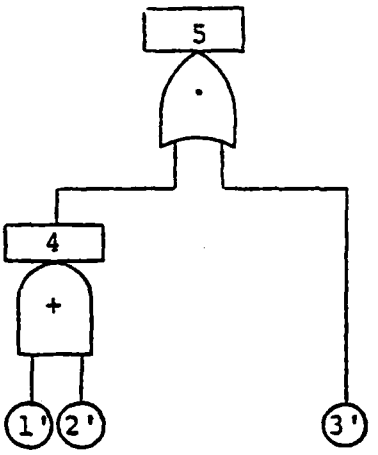
Hence, it is important to find a way of applying our procedure for minimizing the expected cost of testing a system which is presented as a fault tree. To do this we suggest using the decomposition idea, as defined in Section 1.1.9. This set of rules is by no means the optimal procedure, but it seem to work well in many cases.

To ease the demonstration, consider fault trees which do not have replicated basic events, and only two types of "gate" events: "OR" and "AND" gates. Gate events have inputs, which can be either basic events or other gate events, and their output, or the occurrence of the gate event, depends on the occurrence of the input events together with the boolean function that the gate event represents.

The "OR" gate is the fault tree equivalent to a series coherent system, and its output event occurs if one or more of its input events which are the failures of the components occur. The "AND" gate is the fault tree equivalent to a parallel system, and its output occurs if and only if all its input events occur. Thus, the structure function: $\phi(\underline{X}) = 1 - [(1 - x_1x_2)(1 - x_3)]$ with 3 components (3 basic events) can be presented as a coherent system:



or as a fault tree, with 2 gate events:



where:

(i) 1',2',3' are basic events which are the complementary events of 1,2,3 in the above coherent system.

(ii) 4 is a gate event, and "+" represents an OR gate.

(iii) 5 is a gate event, and "·" represents an AND gate.

Note that, as we have already discussed before evaluating the reliability of the coherent system gives us its probability to work. Evaluating the fault tree gives us the probability of a system's failure.

In the context of coherent systems, a module is a subset of components which behaves like a "supercomponent," i.e., effects the system's operation only through the operation of the "supercomponent." If a gate event has inputs which do not appear anywhere else in the tree, then

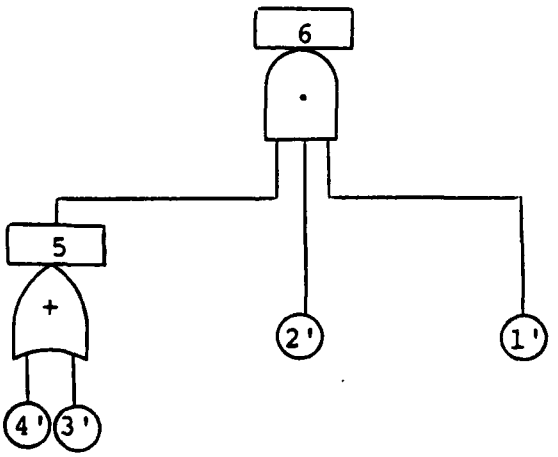
the effect that these inputs have on the top event, is only through the output event of this gate. This is analogous to the module in coherent systems which we discussed above and defined in 1.1.9.

The problem of minimizing the cost of testing systems which are represented as fault trees can now be handled. Consider first all the gate events which have only basic events as inputs. Since the assumption was made that there are no replicated events in the tree, these gate events represent modules of the system, which correspond either to series or parallel coherent systems. We use the optimal procedures of Sections 2.1 and 2.2 respectively to get the optimal order of testing each of these modules, the expected cost of the test, and the probability of functioning for each of the modules. Replacing the gate events under consideration with "superevents," with computed costs and probabilities, results in having new gate events with only basic events or superevents as their inputs. Continuing with the same procedure, we will eventually reach the situation where the top event has only basic events or superevents as its inputs. The top event can now be represented as a series or parallel system, and it is easy to find which of its input events should be tested first. Since only basic events can be tested, if the event that was indicated is a superevent, we have to look at its input and so on, until a basic event is found

to be the first event which should be tested. Unfortunately, example (ii) in Section 3.3 demonstrates that after finding the first component to be tested, the optimal procedure does not necessarily continue to test the components in the same module, until it determines the state of the module. Instead, after getting the result of the first test, we have to recalculate the expected cost and the probability of success for the remainder of the module, and update the costs and probabilities for all the modules of which this module is a submodule. After doing this all the way until the top event, the next basic event to be tested can be determined, and the process continues in this fashion. We demonstrate the operation by an example.

Example:

Consider Example 2.3.3 and construct its corresponding fault tree:

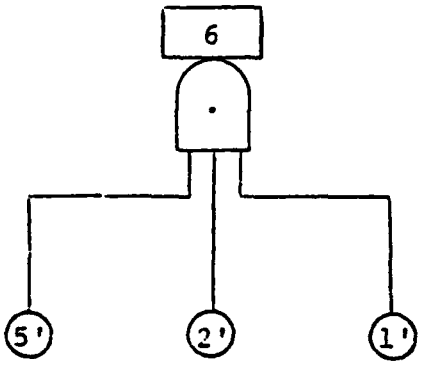


with the following information on basic events:

i	c_i	p_i'
1'	30	.5
2'	15	.75
3'	12	.4
4'	10	.5

Iteration #1:

Event 5 is a gate event which corresponds to a series system, so it can be replaced by a superevent after calculating its probability to function, and its expected cost of testing. For event 5 the inputs are events 3' and 4' and since $\frac{c_4}{p_4'} = 20 < 30 = \frac{c_3}{p_3'}$ the optimal order of testing is to test first event 4', and then event 3'. The expected cost of the test is: $c_4 + (1 - p_4')c_3 = 10 + .5 \cdot 12 = 16$, and the probability that gate event 5 will contribute to the failure of the top event is: $.4 + .5 - .2 = .7$. Replacing gate event 5 by a superevent results in a modified tree:



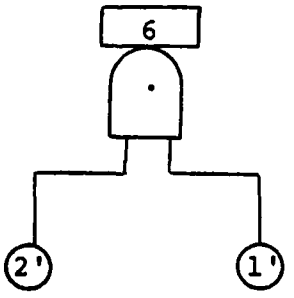
where:

i	c_i	p_i'
1'	30	.5
2'	15	.75
5'	16	.7

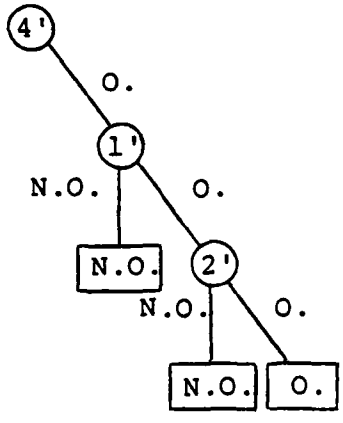
Since this tree represents a parallel system, and since: $\frac{16}{.3} < \frac{15}{.25} = \frac{30}{.5}$ $\left(\frac{c_5}{1 - p_5'} < \frac{c_2}{1 - p_2'} = \frac{c_1}{1 - p_1'}\right)$ we must test event 5' first. Event 5' is a superevent, but we have already seen that the first basic event that should be tested is event 4'. Therefore, this is the first component to be tested, and we now proceed to iteration #2.

Iteration #2:

If event 4' occurred, i.e., the 4^{th} component has failed, the gate event 5 also occurred, since the path-set that includes the 4^{th} component has failed. For the top event to occur, the occurrence of all its immediate inputs is required, so we still need to check events 1' and 2'. The fault tree now is:



and since $\frac{30}{.5} = \frac{15}{.25}$, either one of components 1' or 2' can be tested next. Depending on the occurrence of the tested event, either the last component needs to be checked, or the determination of the nonoccurrence of the checking procedure completed so far as a search tree is given by:



where:

O. stands for occurrence

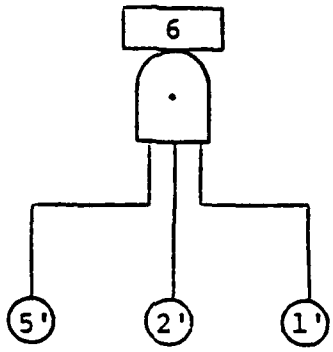
N.O. stands for nonoccurrence

1', 2', 4' are basic events.

Iteration #3:

If event 4' did not occur, i.e., component 4 was found to be working, the modification of the remaining cost and probability of superevent 5 is needed. Since superevent 5

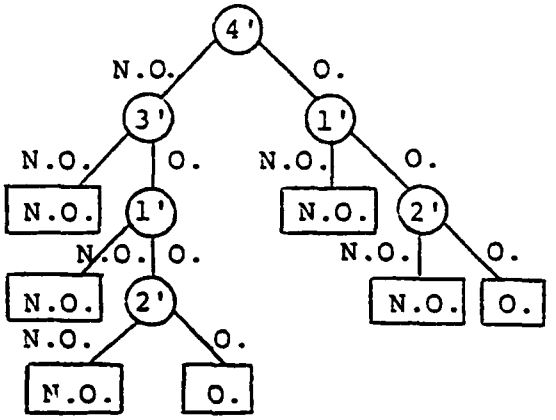
can now occur only if event 3' occurs, the probability of its occurrence is p_3' and the cost of its testing is c_3. The new tree under consideration is now:



where

i	c_i	p_i'
1'	30	.5
2'	15	.75
5'	12	.4

This tree corresponds to a series system, and since: $\frac{12}{.6} < \frac{15}{.25} = \frac{30}{.5}$ the optimal order of testing is: event 3' first and then either 1' or 2'. The final checking procedure can be represented by the following testing tree:



and the expected cost of testing, using this procedure is:

$$C = c_4 + (1 - p_4')c_3 + [p_4' + (1 - p_4')p_3']c_1 + [p_4'p_1' + (1 - p_4')p_3'p_1']c_2$$

$$= 30 + 6 + 21 + 1.5 = 58.5 \ .$$

4.3 Minimizing the Expected Cost of Testing the Components when the System's State is Known

Another important question, which is different from the one we dealt with throughout this dissertation but is similar in its nature, is how to identify efficiently the failed components, when the system is known to be failed. Solving this problem will also provide us with an answer to the dual question which is how to identify the working components, when the system is known to be working.

We present here some preliminary results which we obtained for the simple cases of series and parallel systems,

and the parallel-series and series-parallel structures. A further investigation is needed to give a solution to this problem for the general structure of coherent systems.

Introducing somewhat different notations:

p_i^* = Conditional probability that the i^{th} component has caused the system to fail, given that the system has failed. Obviously, $\sum_{i=1}^{n} p_i^* = 1$.

c_i = Cost of testing the i^{th} component.

$C(\underline{p}^*)$ = Minimum expected cost of testing the system, given the vector of conditional probabilities: $\underline{p}^*$.

$\underline{p}^{*,i}$ = Conditional probabilities vector, where

$$p_j^{*,i} = \begin{cases} \dfrac{p_j^*}{1 - p_j^*} & i \neq j \\ 0 & i = j \end{cases} .$$

We assume that the failure distribution functions of the components are continuous, and therefore the probability that two failures will occur at the same time is 0.

4.3.1 Series Systems

For series systems, the failure of one component causes the failure of the whole system. We want to minimize the expected cost of finding the component that has failed. The minimum expected cost is given by:

$$(4.1) \qquad C(\underline{p}^*) = \min_i \left\{ c_i + (1 - p_i^*) C(\underline{p}^{*,i}) \right\} .$$

Proposition:

The testing procedure: $\pi = (1,2, \ldots, n)$ is optimal for any series system, if and only if:

$$(4.2) \qquad \frac{c_1}{p_1^*} \leq \frac{c_2}{p_2^*} \leq \cdots \leq \frac{c_n}{p_n^*} .$$

Proof:

Consider the procedure which tests component 1 first, then test component 2, and then continue optimally. The expected cost of this procedure is given by:

$$(4.3) \quad C^1(\underline{p}^*) = c_1 + (1 - p_1^*)c_2 + (1 - p_1^*)\left(1 - \frac{p_2^*}{1 - p_1^*}\right) C(\underline{p}^{*,1,2})$$

$$= c_1 + (1 - p_1^*)c_2 + (1 - p_1^* - p_2^*) C(\underline{p}^{*,1,2}) .$$

Now consider the procedure which tests component 2 first, then tests component 1, and then continues optimally. The expected cost of this procedure is given by:

$$C^2(\underline{p}^*) = c_2 + (1 - p_2^*)c_1 + (1 - p_2^*)\left(1 - \frac{p_1^*}{1 - p_2^*}\right) C(\underline{p}^{*,2,1})$$

$$(4.4)$$

$$= c_2 + (1 - p_2^*)c_1 + (1 - p_2^* - p_1^*) C(\underline{p}^{*,2,1}) .$$

AD-A078 824 CALIFORNIA UNIV BERKELEY OPERATIONS RESEARCH CENTER F/G 12/2
OPTIMAL STATE DETECTION POLICIES FOR COHERENT SYSTEMS,(U)
NOV 77 Y BEN-DOV N00014-75-C-0781
UNCLASSIFIED ORC-77-30 NL

END
DATE FILMED
1-80
DDC

Note that $\underline{p}^{*,1,2}$ is the vector of conditional probabilities such that: $p_i^{*,1,2} = \begin{cases} \dfrac{p_i^*}{(1-p_1^*)(1-p_2^*)} & i \neq 1,2 \\ 0 & i = 1 \; ; \; i = 2 \end{cases}$ and

$\underline{p}^{*,2,1}$ is the vector of conditional probabilities such that: $p_i^* = \begin{cases} \dfrac{p_i^*}{(1-p_2^*)(1-p_1^*)} & i \neq 2,1 \\ 0 & i = 2 \; ; \; i = 1 \end{cases}$ so: $\underline{p}^{*,1,2} = \underline{p}^{*,2,1}$

and also: $C(\underline{p}^{*,1,2}) = C(\underline{p}^{*,2,1})$. Therefore:

$$(4.3) \leq (4.4) \iff c_1 + (1 - p_1^*)c_2 \leq c_2 + (1 - p_2^*)c_1$$

$$\iff \frac{c_1}{p_1^*} \leq \frac{c_2}{p_2^*} .$$

We can apply the same considerations to any pair of components in the system, and by doing this complete the proof of the proposition.■

Remark:

Let r_i be the cost of repairing the i^{th} component, $i = 1,2, \ldots, n$. Consider the problem of minimizing the expected cost of restarting a failed series system. The process of restarting the system consists of two stages:

Identifying the failed component and repairing it. The minimum expected cost of restarting a failed series system is given by:

$$(4.5) \quad C(\underline{p}^*, \underline{r}) = \min_i \left\{ c_i + p_i^* r_i + (1 - p_i^*) C(\underline{p}^*, i, \underline{r}) \right\} .$$

It is easily shown that the optimal procedure stays the same, independently of the different repair prices, since we only repair the unique component which is found to be failed.

4.3.2 Parallel Systems

A failure of a parallel system implies that all of its components have failed. Therefore, the question of testing the components is irrelevant to this case, since we know, for sure, that all of them have failed. If we are interested in minimizing the cost of restarting a failed parallel system, we should repair the component with the least repair cost, thus assuring that the system will be functioning again.

4.3.3 Parallel-Series Systems

To solve this case we must assume that whenever a component fails in any of the minimal path sets, all the other components in this path set will enter a mode of suspended operation, and hence will not fail.

If a parallel-series system has failed, we know that all its series subsystems have failed. Let: $R_1, R_2, \ldots, R_r$ be the series subsystems, which are the minimal path sets of the system. Based on the additional assumption, we can calculate the minimum expected cost of testing any of these series subsystems, using (4.1), and the optimal testing procedure given in the proposition. Denote the minimum expected cost of testing the i^{th} path-set by $C(R_i)$. The optimal testing procedure will be to test first the component (according to (4.2)) in the j^{th} path set that satisfies: $R_j = \min_i \{C(R_i)\}$. If this component is found to be failed, we know that this is the only failed component in the j^{th} path-set, and by repairing it we restart the system. Otherwise, if the component is found to be in a working condition (even though it is in suspended operation), we update the vector of conditional probabilities, recompute $C(R_i)$ for all the minimal path sets, and continue in the same manner.

4.3.4 Series-Parallel Systems

A failure of a series-parallel system implies that exactly one of its parallel subsystems, which are the minimal cut sets of the system, is failed. We do not know which of the minimal cut sets has failed, and therefore to get the expression for minimizing the expected time of testing each of these parallel subsystems, we must use

(2.2) and the optimal procedure for testing parallel systems, when the state of the system is unknown. Let $K_1, \ldots, K_2$ be the minimal cut sets of the system, and denote by $C(K_j)$ $j = 1,2, \ldots, n$ the expected cost of testing the j^{th} cut set as given by (2.2).

The conditional probability that any of these cut-sets has failed, given that the system has failed, is the sum of the conditional probabilities of the components in each one of the minimal cut sets, to be denoted by $P(K_j)$ $j = 1,2, \ldots, n$. So, to start our testing procedure we choose the j^{th} minimal cut set that satisfies: $\frac{C(K_j)}{P(K_j)} = \min_i \left\{ \frac{C(K_i)}{P(K_i)} \right\}$, and test the first component in this cut-set, according to the optimal procedure for parallel systems. After the test we update the vector of conditional probabilities and continue with a series-parallel system with $(n - 1)$ components, using the same set of rules.

REFERENCES

[1] Barlow, R. E. and F. Proschan, MATHEMATICAL THEORY OF RELIABILITY, John Wiley and Sons, New York, (1965).

[2] Barlow, R. E. and F. Proschan, STATISTICAL THEORY OF RELIABILITY AND LIFE TESTING, Holt, Rinehart and Winston, New York, (1975).

[3] Barlow, R. E. and F. Proschan, "Importance of System Components and Fault Events," ORC 74-3, Operations Research Center, University of California, Berkeley, California, (1974).

[4] Barlow, R. E. and H. Lambert, "Introduction to Fault Tree Analysis," in RELIABILITY AND FAULT TREE ANALYSIS, SIAM, Pennsylvania, (1975).

[5] Barlow, R. E., "Coherent Systems with Multi-State Components," ORC 77-5, Operations Research Center, University of California, Berkeley, California, (1977).

[6] Birnbaum, Z. W. and J. D. Esary, "Modules of Coherent Binary Systems," SIAM Journal on Applied Mathematics, Vol. 13, pp. 444-462, (1965).

[7] Birnbaum, Z. W., J. D. Esary and S. C. Saunders, "Multi-Component Systems and Structures and Their Reliability," Technometrics, Vol. 3, pp. 55-77, (1961).

[8] Butterworth, R. W., "Modules of Coherent Systems and Their Relationship to Blocking Systems," ORC 69-11, Operations Research Center, University of California, Berkeley, California, (1969).

[9] Butterworth, R. W., "A Set Theoretic Treatment of Coherent Systems," SIAM Journal on Applied Mathematics, Vol. 22, pp. 590-598, (1972).

[10] Butterworth, R. W., "Some Reliability Fault-Testing Models," Operations Research, Vol. 20, pp. 335-343, (1972).

[11] Esary, J. D. and F. Proschan, "Coherent Structures with Non-Identical Components," Technometrics, Vol. 5, pp. 191-209, (1963).

[12] Halpern, J., "Fault Testing for a k-out-of-n System," Operations Research, Vol. 22, pp. 1267-1271, (1974).

[13] Halpern, J., "A Sequential Testing Procedure for a System's State Identification," IEEE Transactions on Reliability, Vol. R-23, No. 4, pp. 267-273, (1974).

[14] Harnisch, H., P. J. Hilton and W. M. Hirsch, "Algebraic and Combinatorial Aspects of Coherent Structures," Transactions New York Academy Sciences, II, 31, pp. 1024-1037, (1969).

[15] Knuth, D. E., THE ART OF COMPUTER PROGRAMMING, Addison-Wesley Publishing Company, Inc., pp. 393-403, (1973).

[16] Lambert, H., "Fault Trees for Decision Making in Systems Analysis," Lawrence Livermore Laboratory Report, UCRL-51829, Livermore, California, (1975).

[17] Lambert, H. and G. Yadigaroglu, "Fault Trees for Diagnosis of System Fault Conditions," Nuclear Science and Engineering, Vol. 62, pp. 30-34, (1977).

[18] Little, J. D. C., K. G. Murty, D. W. Sweeney and C. Karel, "An Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 11, pp. 972-989, (1963).

[19] Ross, S. M., "Multicomponent Reliability Systems," ORC 74-4, Operations Research Center, University of California, Berkeley, California, (1974).

[20] Slage, J. R., "An Efficient Algorithm for Finding Certain Minimum-Cost Procedures for Making Binary Decisions," Journal of the ACM, Vol. 11, pp. 253-264, (1964).